## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tableau Server Enterprise Deployment Guide</td>
<td>1</td>
</tr>
<tr>
<td>Who should read this</td>
<td>1</td>
</tr>
<tr>
<td>Version</td>
<td>2</td>
</tr>
<tr>
<td>Highlight features</td>
<td>2</td>
</tr>
<tr>
<td>Licensing</td>
<td>3</td>
</tr>
<tr>
<td><strong>Part 1 - Understanding Enterprise Deployment</strong></td>
<td>4</td>
</tr>
<tr>
<td>Industry standards and deployment requirements</td>
<td>4</td>
</tr>
<tr>
<td>Security measures</td>
<td>5</td>
</tr>
<tr>
<td>Web proxy tier</td>
<td>6</td>
</tr>
<tr>
<td>Load-balancers</td>
<td>6</td>
</tr>
<tr>
<td>Application tier</td>
<td>7</td>
</tr>
<tr>
<td>Data tier</td>
<td>7</td>
</tr>
<tr>
<td><strong>Part 2 - Understanding the Tableau Server Deployment Reference Archi</strong></td>
<td>8</td>
</tr>
<tr>
<td>Tableau Server Processes</td>
<td>9</td>
</tr>
<tr>
<td>PostgresSQL Repository</td>
<td>10</td>
</tr>
<tr>
<td>Node 1: Initial node</td>
<td>10</td>
</tr>
<tr>
<td>Node 1 failover and automated restoration</td>
<td>11</td>
</tr>
<tr>
<td>Nodes 1 and 2: Application servers</td>
<td>11</td>
</tr>
<tr>
<td>Scaling application servers</td>
<td>12</td>
</tr>
<tr>
<td>Nodes 3 and 4: Data servers</td>
<td>13</td>
</tr>
</tbody>
</table>
Tableau Server Enterprise Deployment Guide

Install initial node of Tableau Server ......................................................... 38
Run installation package and initialize TSM .................................................. 38
Activate and register Tableau Server ............................................................ 40
Configure identity store .................................................................................. 41
Configure external Postgres .......................................................................... 41
Finish Node 1 installation ............................................................................... 42
Verification: Node 1 configuration ................................................................. 43
Take Step 2 tar backups ................................................................................. 44
Install Tableau Server on remaining nodes .................................................. 48
Generate, copy, and use the bootstrap file to initialize TSM ......................... 50
Configure processes ....................................................................................... 51
  Configure Node 2 ......................................................................................... 51
  Configure Node 3 ......................................................................................... 52
Deploy coordination service ensemble to Nodes 1-3 ..................................... 53
Take Step 3 tar backups ................................................................................. 54
Configure Node 4 .......................................................................................... 58
Final process configuration and verification ................................................. 58
Perform backup ............................................................................................. 59

Part 5 - Configuring Web Tier ....................................................................... 61
Authentication and authorization ................................................................. 62
  Pre-authentication with an AuthN module .................................................. 62
Configuration overview .................................................................................. 63
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable SAML on Tableau Server for IdP</td>
<td>80</td>
</tr>
<tr>
<td>Validate SAML functionality</td>
<td>82</td>
</tr>
<tr>
<td>Validation troubleshooting</td>
<td>83</td>
</tr>
<tr>
<td><strong>Part 6 - Post-Installation Configuration</strong></td>
<td>84</td>
</tr>
<tr>
<td>Configure SSL/TLS from load balancer to Tableau Server</td>
<td>84</td>
</tr>
<tr>
<td>Example: Configure SSL/TLS in AWS reference architecture</td>
<td>84</td>
</tr>
<tr>
<td>Step 1: Generate a self-signed certificate</td>
<td>85</td>
</tr>
<tr>
<td>Step 2: Configure proxy server for SSL</td>
<td>86</td>
</tr>
<tr>
<td>Step 3: Configure Tableau Server for external SSL</td>
<td>89</td>
</tr>
<tr>
<td>Step 4: Optional authentication configuration</td>
<td>89</td>
</tr>
<tr>
<td>Step 5: Configure AWS load balancer for HTTPS</td>
<td>89</td>
</tr>
<tr>
<td>Step 6: Verify SSL</td>
<td>91</td>
</tr>
<tr>
<td>Configure SSL for Postgres</td>
<td>91</td>
</tr>
<tr>
<td>Verify SSL connectivity</td>
<td>93</td>
</tr>
<tr>
<td>Configure SMTP and event notifications</td>
<td>94</td>
</tr>
<tr>
<td>Install PostgreSQL driver</td>
<td>96</td>
</tr>
<tr>
<td>Configure strong password policy</td>
<td>97</td>
</tr>
<tr>
<td><strong>Part 7 - Tools and Troubleshooting</strong></td>
<td>99</td>
</tr>
<tr>
<td>Initial node automated recovery</td>
<td>99</td>
</tr>
<tr>
<td>Troubleshooting initial node recovery</td>
<td>101</td>
</tr>
<tr>
<td>Rebuilding the failed node</td>
<td>101</td>
</tr>
<tr>
<td>switchto</td>
<td>101</td>
</tr>
</tbody>
</table>
Appendix - AWS Deployment Toolbox .........................................................105
EDGSetup automated installation script ..................................................105
Tableau Server Enterprise Deployment Guide

The Tableau Server Enterprise Deployment Guide (EDG) has been developed to provide prescriptive guidance for deploying Tableau Server (on-premises or in the cloud). The Guide provides deployment guidance for enterprise scenarios in context of a reference architecture. We have tested the reference architecture to verify compliance with security, scale, and performance benchmarks, which conform to industry-standard best practices.

At a high-level, the core features of an industry standard enterprise deployment consist of a tiered topology where each layer of server application functionality (web gateway tier, application tier, and data tier) is bound and protected by access-controlled subnets. Users accessing the server application from the internet are authenticated at the web tier. Once authenticated, the request is proxied to a protected subnet where the application tier handles the business logic. High-value data is protected by the third subnet: the data tier. Services in the application tier communicate over the protected network to the data tier to service data requests to the backend data sources.

In this deployment, security is at the forefront of all design decisions and implementation. However, reliability, performance, and scalability are also priority requirements. Given the distributed and modular design of the reference architecture, reliability and performance scale in a linearly predictable way by strategically co-locating compatible services at each node and adding services at chokepoints.

Who should read this

The EDG has been developed for enterprise IT administrators who may require:

- An IT-managed Tableau deployment
- Industry compliance enforcement
Industry deployment best practices
Secure deployment by default

The EDG is an implementation guide for deploying the enterprise reference architecture. While this version of the EDG includes an example AWS/Linux implementation, the Guide can be used as a resource by experienced enterprise IT administrators to deploy the prescribed reference architecture into any industry standard data center environment.

Version

This version of EDG was developed for the 2021.2.3 version (or later) of Tableau Server. While you may use the EDG as a general reference for deploying older versions of Tableau Server, we recommend that you deploy the reference architecture with Tableau Server 2021.2.3 or later. Some features and options are not available on older versions of Tableau Server.

For the most up-to-date features and improvements, we recommend deploying EDG with Tableau Server 2022.1 and later.

The reference architecture described in this Guide supports the following Tableau clients: Web authoring with compatible browsers, Tableau Mobile, and Tableau Desktop version 2021.2.1 or later. Other Tableau clients (Tableau Prep, Bridge, etc) have not yet been validated with the reference architecture.

Highlight features

The first version of the Tableau Server reference architecture introduces the following scenarios and features:

- Client pre-authentication: Tableau clients (Desktop, Mobile, Web Authoring) authenticate with the corporate authentication provider in the web tier before accessing internal Tableau Server. This process is managed by configuring an authN plug-in on the reverse proxy server. See Web proxy tier.
Tableau Server Enterprise Deployment Guide

- Zero trust deployment: Because all traffic to Tableau Servers is pre-authenticated, the entire Tableau deployment operates in a private subnet that does not require a trusted connection.
- External repository: The reference architecture specifies installing the Tableau repository onto an external PostgreSQL database, allowing DBAs to manage, optimize, scale, and back up the repository as a generic database.
- Initial node recovery: The EDG introduces a script that automates initial node restoration in the event of a failure.
- Tar-based backup and restore: Use familiar tar backups at strategic milestones of the Tableau deployment. In the event of a failure or deployment misconfiguration, you can quickly recover to the previous deployment stage by recovering the associated tar backup.
- Performance improvement: Customer and lab validation show a 15-20% performance improvement when running EDG compared to standard deployment.

Licensing

The Tableau Server reference architecture prescribed in this Guide requires a Server Management Add-on license to enable Tableau Server External Repository. You may also optionally deploy Tableau Server External File Store, which also requires the Server Management Add-on License. See About Tableau Server Management Add-on (Linux).
**Part 1 - Understanding Enterprise Deployment**

Part 1 describes, in more detail, the features and requirements of industry-standard enterprise deployment for which the Tableau Server Enterprise Deployment Guide has been designed.

The following network diagram shows a generic datacenter tiered deployment with Tableau Server reference architecture.

### Industry standards and deployment requirements

The following are features of industry standard deployments. These are the requirements that the reference architecture has been designed for:

- A multi-tiered network design: The network is bound by protected subnets to limit access at each layer: web layer, application layer, and data layer. No single communication is able to pass across subnets, as all communication is terminated at the next subnet.
Ports and protocols blocked by default: Each subnet or security group will block all inbound and outbound ports and protocols by default. Communication is enabled, in part, by opening exceptions in the port or protocol configuration.

Off-box web authentication: User requests from the internet are authenticated by an authentication module on the reverse proxy in the web tier. Therefore, all requests to the application layer are authenticated at the web tier before passing into the protected application layer.

Platform-independent: Solution can be deployed with on-premises server applications or in the cloud.

Technology-agnostic: Solution can be deployed in a virtual machine environment or in containers. May also be deployed on Windows or Linux. However, this initial version of the reference architecture and supporting documentation has been developed for Linux running in AWS.

Highly available: All components in the system are deployed as a cluster and designed to operate in an active/active or active/passive deployment.

Siloed roles: Each server performs a discrete role. This design partitions all servers such that access may be minimized to service-specific administrators. For example, DBAs manage PostgreSQL for Tableau, identity administrators manage authentication module in web tier, network and cloud administrators enable traffic and connectivity.

Linearly scalable: as discrete roles, you can scale each tier service independently according to load profile.

Client support: The reference architecture supports all Tableau clients: Tableau Desktop (versions 2021.2 or later), Tableau Mobile, and Tableau Web Authoring.

Security measures

As stated, a primary feature of industry standard datacenter design is security.

Access: Each tier is bound by a subnet that enforces access control at the network layer using port filtering. Communication access between subnets may also be enforced by the application layer with authenticated services between processes.

Integration: Architecture is designed to plug-in with Identity Provider (IdP) on reverse proxy in the web tier.

Privacy: Traffic into the web tier is encrypted from the client with SSL. Traffic into the internal subnets may optionally be encrypted as well.
Web proxy tier

The web tier is a subnet in the DMZ (also referred to as the perimeter zone) that acts as a security buffer between the internet and the internal subnets where applications are deployed. The web tier hosts reverse proxy servers that do not store any sensitive information. The reverse proxy servers are configured with an AuthN plugin to pre-authenticate client sessions with a trusted IdP, before redirecting the client request to Tableau Server. For more information, see Pre-authentication with an AuthN module.

Load-balancers

The deployment design includes an enterprise load-balancing solution in front of the reverse proxy servers.

Load balancers provide important security and performance enhancements, by

- Virtualizing the front-end URL for the application tier services
- Enforcing SSL encryption
- Offloading SSL
- Enforcing compression between the client and the web tier services
- Protecting against DOS attacks
- Providing high-availability

**Note:** Tableau Server version 2022.1 includes the Tableau Server Independent Gateway. The Independent Gateway is a standalone instance of the Tableau Gateway process that serves as a Tableau-aware reverse proxy. At the time of release, the Independent Gateway has been validated, but not fully tested in the EDG reference architecture. After full testing is complete the EDG will be updated with Tableau Server Independent Gateway prescriptive guidance.
Application tier

The application tier is in a subnet that runs the core business logic of the server application. The application tier consists of services and processes that are configured across distributed nodes in a cluster. The application tier is only accessible from the web tier and is not directly accessible by users.

Performance and reliability are improved by configuring the application processes such that processes with different resource-use profiles (i.e., CPU intensive vs memory intensive) are co-located.

Data tier

The data tier is a subnet that holds valuable data. All traffic to this tier originates from the application tier and is therefore already authenticated. In addition to access requirements at the network layer with port configuration, this layer should include authenticated access and optionally encrypted traffic with the application tier.
Part 2 - Understanding the Tableau Server Deployment Reference Architecture

The following image shows the relevant Tableau Server processes and how they are deployed in the reference architecture. This deployment is considered the minimal enterprise-appropriate Tableau Server deployment.

The process diagrams in this topic are intended to show the major, defining processes of each node. There are many supporting processes that also run on the nodes that are not shown in the diagrams. For a list of all processes, see the configuration section of this guide, Part 4 - Installing and Configuring Tableau Server.
Tableau Server Processes

The Tableau Server reference architecture is a four node Tableau Server cluster deployment with external repository on PostgreSQL:

- Tableau Server initial node (Node 1): Runs required TSM administrative and licensing services that can only be run on a single node in the cluster. In the enterprise context, the Tableau Server initial node is the primary node in the cluster. This node also runs redundant application services with Node 2.
- Tableau Server application nodes (Node 1 and Node 2): The two nodes serve client requests, connect to and query data sources and to the data nodes.
- Tableau Server data nodes (Node 3 and Node 4): Two nodes that are dedicated to managing data.
- External PostgreSQL: this host runs the Tableau Server Repository process. For HA deployment you must run an additional PostgreSQL host for active/passive redundancy.

You can also run PostgreSQL on Amazon RDS. For more information about the differences between running the repository on RDS vs an EC2 instance, see Tableau Server External Repository (Linux).

Deploying Tableau Server with an external repository requires a Server Management Add-on license.

If your organization does not have in-house DBA expertise, you may optionally run the Tableau Server Repository process in the default, internal PostgreSQL configuration. In the default scenario, the Repository is run on a Tableau node with embedded PostgreSQL. In this case, we recommend running the Repository on a dedicated Tableau node, and a passive Repository on an additional dedicated node to support Repository failover. See Repository Failover (Linux).

By way of example, the AWS implementation described in this Guide explains how to deploy the external repository on PostgreSQL running on an EC2 instance.

- Optional: If your organization uses external storage, you may deploy the Tableau File
Store as an external service. This Guide does not include the External File Store in the core deployment scenario. See *Install Tableau Server with External File Store (Linux)*.

Deploying Tableau Server with an external File Store requires a Server Management Add-on license.

**PostgreSQL Repository**

Tableau Server Repository is a PostgreSQL database that stores server data. This data includes information about Tableau Server users, groups and group assignments, permissions, projects, data sources, and extract metadata and refresh information.

The default PostgreSQL deployment consumes almost 50% of system memory resources. Based on its usage (for production and large production deployments) resource usage can go up. For this reason, we recommend running the Repository process on a computer that is not running any other resource-intensive server components like VizQL, Backgrounder, or Data Engine. Running the Repository process along with any of these components will create IO contentions, resource constraint, and will degrade overall performance of the deployment.

**Node 1: Initial node**

The initial node runs a small number of important processes and shares the application load with Node 2.

The first computer you install Tableau on, the "initial node," has some unique characteristics. Three processes run only on the initial node and cannot be moved to any other
node except in a failure situation, the License Service (License Manager), Activation Service, and TSM Controller (Administration Controller).

Node 1 failover and automated restoration

Each of these services are critical to the health of a Tableau Server deployment. In the event of a Node 1 failure, users will still be able to connect to the Tableau Server deployment, as a properly configured reference architecture will route requests to Node 2. However without these core services, the deployment will be in a critical state of pending failure. See Initial node automated recovery.

Nodes 1 and 2: Application servers
Nodes 1 and 2 run the Tableau Server processes that serve client requests, query data sources, generate visualizations, handle content and administration, and other core Tableau business logic. The application servers do not store user data.

**Note:** "Application Server" is a term that also refers to a Tableau Server process that is listed in TSM. The underlying process for "Application Server" is VizPortal.

Run in parallel, Node 1 and Node 2 scale to service requests from the load-balancing logic run on the reverse proxy servers. As redundant nodes, should one of these nodes fail, then client requests and servicing are handled by the remaining node.

The reference architecture has been designed so that complimentary application processes run on the same computer. This means the processes are not competing for computing resources and creating contention.

For example, VizQL, a core processing service on application servers, is highly CPU and memory-bound, VizQL uses almost 60-70% of the CPU and memory on the computer. For this reason, the reference architecture is designed so that no other memory or CPU-bound processes are on the same node as VizQL. Testing shows that the amount of the load or number users doesn’t affect the memory or CPU usage on VizQL nodes. For example, reducing the number of concurrent users in our load test only effects the performance of the dashboard or the visualization loading process, but does not reduce resource utilization. Therefore, based on the available memory and CPU during peak usage, you may consider adding more VizQL processes. As a starting place for typical workbooks, allocate 4 cores per VizQL process.

### Scaling application servers

The reference architecture is designed for scale based on a use-based model. As a general starting point, we recommend a minimum of two application servers, each supporting up to 1000 users. As user base increases, plan to add an application server for each additional
1000 users. Monitor usage and performance to tune the user base per host for your organization.

Nodes 3 and 4: Data servers

The File Store, Data Engine (Hyper), and Backgrounder processes are co-located on Nodes 3 and 4 for the following reasons:

- Extract optimization: Running Backgrounder, Hyper, and File Store on the same node optimizes performance and reliability. During the extraction process, Backgrounder queries the target database, creates the Hyper file on the same node, and then uploads to File Store. By co-locating these processes on the same node the extraction creation workflow does not require copying amounts of data across the network or the nodes.

- Complimentary resource balancing: Backgrounder is mainly CPU intensive. Data Engine is a memory-intensive process. Coupling these processes allows maximum resource utilization on each node.

- Consolidation of data processes: Since each of these processes are back-end data processes, it makes sense to run them in the most secure data tier. In future versions of the reference architecture, the application and data servers will run in separate tiers. However, due to application dependencies in the Tableau architecture, application and data servers must run in the same tier at this time.
Scaling data servers

As with application servers, planning the resources that are required for Tableau data servers requires use-based modeling. In general, assume each data server can support up to 2000 extract refresh jobs per day. As your extract jobs increase, add additional data servers without the File Store service. Generally, the two-node data server deployment is suitable for deployments that use the local filesystem for the File Store service. Note that adding more application servers does not impact performance or scale on data servers in a linear fashion. In fact, with the exception of some overhead from additional user queries, the impact of adding more application hosts and users is minimal.
Part 3 - Preparing for Tableau Server Enterprise Deployment

Part 3 describes the requirements for preparing your infrastructure to deploy the Tableau Server reference architecture. Before you begin, we recommend reviewing, Part 2 - Understanding the Tableau Server Deployment Reference Architecture.

In addition to requirement descriptions, this topic provides an implementation example of the reference architecture in an AWS environment. The remainder of this Guide builds on the AWS reference architecture example started in this topic.

A core principle of the reference architecture is standardization with data center security best practices. Specifically, the architecture is designed to segregate services into protected network subnets. Inter-subnet communication is restricted to specific protocol and port traffic.

The following diagram illustrates the reference architecture subnet design for an on-premises deployment or a customer-managed cloud deployment. For an example cloud deployment, see the section below, Example: Configure subnets and security groups in AWS.
Subnets

Create three subnets:

- A web tier
- An application tier
- A data subnet.

Firewall/Security group rules

The tabs below describe the firewall rules for each tier of the datacenter. For AWS-specific security group rules, see the section later in this topic.

Web tier

The web tier is a public DMZ subnet that will handle inbound HTTPS requests and proxy the requests to the application tier. This design provides a layer of defense from malware that may be targeted at your organization. The web tier blocks access to the application/data tier.

<table>
<thead>
<tr>
<th>Traffic</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Bastion subnet (for cloud deployments)</td>
</tr>
<tr>
<td>Inbound</td>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>Internet (0.0.0.0/0)</td>
</tr>
<tr>
<td>Inbound</td>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>Internet (0.0.0.0/0)</td>
</tr>
<tr>
<td>Outbound</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td></td>
</tr>
</tbody>
</table>

Application tier

The application subnet is where the Tableau Server deployment resides. The application
Tableau Server Enterprise Deployment Guide

The application subnet includes the Tableau application servers (Node 1 and Node 2). The Tableau application servers process user requests to the data servers and run core business logic.

The application subnet also includes the Tableau data servers (Node 3 and Node 4).

All client traffic to the application tier is authenticated at the web tier. Administrative access to the application subnet is authenticated and routed through the bastion host.

<table>
<thead>
<tr>
<th>Traffic</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Bastion subnet (for cloud deployments)</td>
</tr>
<tr>
<td>Inbound</td>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>Web tier subnet</td>
</tr>
<tr>
<td>Outbound</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td></td>
</tr>
</tbody>
</table>

Data tier

The application subnet is where the external PostgreSQL database server resides.

<table>
<thead>
<tr>
<th>Traffic</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Bastion subnet (for cloud deployments)</td>
</tr>
<tr>
<td>Inbound</td>
<td>PostgreSQL</td>
<td>TCP</td>
<td>5432</td>
<td>Public tier subnet</td>
</tr>
<tr>
<td>Outbound</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td></td>
</tr>
</tbody>
</table>

Bastion

Most enterprise security teams do not allow direct communication from the on-premises administrative system to the nodes deployed in the cloud. Instead, all administrative SSH traffic to the cloud nodes is proxied through a bastion host (also referred to as a "jump
server”). For cloud deployments, we recommend bastion host proxy connection to all resources in the reference architecture. This is an optional configuration for on-premises environments.

The bastion host authenticates administrative access and only allows traffic over SSH protocol.

<table>
<thead>
<tr>
<th>Traffic</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Admin computer IP address</td>
<td></td>
</tr>
<tr>
<td>Outbound</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td></td>
<td>Web tier subnet</td>
</tr>
<tr>
<td>Outbound</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td></td>
<td>Application tier subnet</td>
</tr>
</tbody>
</table>

**Example: Configure subnets and security groups in AWS**

This section provides step-by-step procedures to create and configure the VPC and network environment for the Tableau Server reference architecture deployment in AWS.

The slides below show the reference architecture in four layers. As you progress through the slides, component elements are layered onto the topology map:

1. VPC subnet topology and EC2 instances: one bastion host, two reverse proxy servers, four Tableau servers, and at least one PostgreSQL server.
2. Protocol flow and internet connectivity. All inbound traffic is managed through the AWS internet gateway. Traffic to the internet is routed through the NAT.
3. Availability zones. The proxy, Tableau Server, and PostgreSQL hosts are evenly deployed across two Availability Zones.
4. Security groups. Four security groups (Public, Private, Data, and Bastion) protect each tier at the protocol level.
Tableau Server Enterprise Deployment Guide

AWS reference architecture

Slide 1: VPC subnet topology and EC2 instances
Slide 2: Protocol flow and connectivity
Slide 3: Availability zones

Tableau Server Enterprise Deployment Guide

Tableau Software
Slide 4: Security groups

AWS Availability Zones and high availability

The reference architecture as presented in this Guide specifies a deployment that provides availability through redundancy when any single host fails. However, in the AWS case where the reference architecture is deployed across two Availability Zones, availability is compromised in the very rare case where an Availability Zone fails.

VPC configuration

This section describes how to:

- Install and configure the VPC
- Configure internet connectivity
Configure VPC

The procedure in this section maps to the UI in the "classic" VPC Experience. You can toggle the UI to display the classic view by turning off the New VPC Experience in the upper-left corner of the AWS VPC Dashboard.

Run VPC wizard to create default Private and Public subnets and default routing and network ACL.

1. Before you configure a VPC, you must create an Elastic IP. Create an allocation using all defaults.
2. Run VPC wizard > "VPC with Public and Private Subnets"
3. Accept most defaults. Except for the following:
   - Enter a VPC name.
   - Specify the Elastic IP Allocation ID.
   - Specify the following CIDR masks:
     - Public subnet's IPv4 CIDR: 10.0.1.0/24, rename this subnet to Public-a.
     - Private subnet's IPv4 CIDR: 10.0.30.0/24, rename this subnet to Private-a.
   - Availability Zone: for both subnets, select the a option for the region that you are in.

   **Note:** For the purposes of this example, we use a and b to distinguish between Availability Zones in a given AWS datacenter. In AWS, Availability Zone names may not match the examples shown here. For example, some Availability Zones include c and d zones within a datacenter.

4. Click **Create VPC**.
5. After VPC is created, create Public-b, Private-b, Data, and Bastion subnets. To create a subnet, click **Subnets > Create subnet**.
- **Public**: For Availability Zone, select the **b** option for the region that you are in. CIDR block: 10.0.2.0/24
- **Private**: For Availability Zone, select the **b** option for the region that you are in. CIDR block: 10.0.31.0/24
- **Data**: For Availability Zone, select the **a** zone for the region that you are in. CIDR block: 10.0.50.0/24. Optional: If you plan to replicate the external database across a PostgreSQL cluster, then create a Data-b subnet in Availability Zone **b** with a CIDR block of 10.0.51.0/24.
- **Bastion**: For Availability Zone, select either zone. CIDR block: 10.0.0.0/24

6. After the subnets are created, edit the route tables on the Public and the Bastion subnets to use the route table that is configured for the associated internet gateway (IGW). And edit the Private and Data subnets to use the route table that is configured for the network address translator (NAT).
   - To determine which route table is configured with the IGW or the NAT, click **Route Tables** in AWS dashboard. Select one of the two route table links to open the property page. Look at the Target value at Routes > Destination > 0.0.0.0/0. The Target value differentiates the type of route and will either start with the `igw-` or `nat-` string.
   - To update route tables, **VPC > Subnets > [subnet_name] > Route table > Edit route table association.**

**Configure security groups**

The VPC wizard creates a single security group that you will not use. Create the following security groups (**Security Groups > Create security group**). The EC2 hosts will be installed into these groups across two Availability Zones as shown in the slide-diagram above.

- Create a new security group: **Private**. This is where all 4 nodes of Tableau Server will be installed. Later in the installation process, the Private security group will be associated with the 10.0.30.0/24 and 10.0.31.0/24 subnets.
- Create a new security group: **Public**. This is where proxy servers will be installed. Later in the installation process, the Public security group will be associated with the 10.0.1.0/24 and 10.0.2.0/24 subnets.
Tableau Server Enterprise Deployment Guide

- Create a new security group: **Data.** This is where the PostgreSQL external Tableau repository will be installed. Later in the installation process, the Data security group will be associated with the 10.0.50.0/24 (and optionally, 10.0.51.0/24) subnet.
- Create a new security group: **Bastion.** This is where you'll install the bastion host. Later in the installation process, the Bastion security group will be associated with the 10.0.0.0/24 subnet.

Specify inbound and outbound rules

In AWS, security groups are analogous to firewalls in an on-prem environment. You must specify the traffic type, (eg, https, https, etc), protocol (TCP or UDP), and ports or port range (eg 80, 443, etc) that are allowed to pass in and/or out of the security group. For each protocol you must also specify the destination or source traffic.

**Public security group rules**

<table>
<thead>
<tr>
<th>Inbound rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>HTTP</td>
</tr>
<tr>
<td>HTTPS</td>
</tr>
<tr>
<td>SSH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outbound rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>All traffic</td>
</tr>
</tbody>
</table>

**Private security group rules**

The Private security group includes an inbound rule to allow HTTP traffic from the Public security group. Allow HTTP traffic only during the deployment process to verify connectivity.
We recommend removing the HTTP inbound rule after you have finished deploying the reverse proxy and configuring SSL to Tableau.

### Inbound rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>Public security group</td>
</tr>
<tr>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>Public security group</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>TCP</td>
<td>5432</td>
<td>Data security group</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Bastion security group</td>
</tr>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>Private security group</td>
</tr>
</tbody>
</table>

### Outbound rule

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>TCP</td>
<td>5432</td>
<td>Data security group</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Bastion security group</td>
</tr>
</tbody>
</table>

### Data security group rules

#### Inbound rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL</td>
<td>TCP</td>
<td>5432</td>
<td>Private security group</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Bastion security group</td>
</tr>
</tbody>
</table>
### Outbound rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>TCP</td>
<td>5432</td>
<td>Private security group</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Bastion security group</td>
</tr>
</tbody>
</table>

### Bastion host security group rules

#### Inbound rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>The IP address and net mask of the computer that you will use to log into AWS (admin computer).</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Private security group</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Public security group</td>
</tr>
</tbody>
</table>

#### Outbound rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>The IP address and net mask of the computer that you will use to log into AWS (admin computer).</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Private security group</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Public security group</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>Data security group</td>
</tr>
<tr>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>0.0.0.0/0 (Optional: create this rule</td>
</tr>
</tbody>
</table>
Enable auto-assign public IP

This provides you with an IP address for connecting to the proxy servers and bastion host.

For Public and Bastion subnets:

1. Select the subnet
2. Under Actions menu, select "Modify auto-assign IP settings."
3. Click "Enable auto-assign public IPv4 addresses."
4. Click Save.

Load balancer

Note: If you are installing into AWS and following the example deployment in this guide, then you should install and configure the AWS load balancer later in the deployment process, as described in Part 5 - Configuring Web Tier.

For on-premises deployments, work with your network administrators to deploy load balancers to support the web tier of the reference architecture:

- A web-facing application load balancer that accepts HTTPS requests from Tableau clients and communicates with the reverse proxy servers.
- Reverse proxy:
  - We recommend a minimum of two proxy servers for redundancy and to handle client load.
  - Receives HTTPS traffic from load balancer.
  - Supports sticky session to Tableau host.
  - Configure proxy for round robin load balancing to each Tableau Server running the Gateway process.
  - Handles authentication requests from external IdP.
Forward proxy: Tableau Server requires access to the internet for licensing and map functionality. Depending on your forward proxy environment, you may need to configure forward proxy safelists for Tableau service URLs. See *Communicating with the Internet* (Linux).

**Install host computers**

**Minimum recommended hardware**

The following recommendations are based on our testing of real-world data in the reference architecture.

**Application servers:**

- CPU: 8 physical cores (16vCPUs),
- RAM: 128 GB (16 GB/physical Core)
- Disk space: 50 GB

**Data servers**

- CPU: 8 physical cores (16vCPUs),
- RAM: 128 GB (16 GB/physical Core)
- Disk space: 1 TB. If your deployment will make use of external storage for the Tableau File Store, you will need calculate the appropriate disk space. See *Install Tableau Server with External File Store* (Linux).

**Proxy servers**

- CPU: 2 physical cores (4vCPUs),
- RAM: 8 GB (4 GB/physical Core)
- Disk space: 100 GB

**External repository database**

- CPU: 8 physical cores (16vCPUs),
- RAM: 128 GB (16 GB/physical Core)
Disk space requirement is dependent on your data load and how that will impact backup. See the section, *Backup and restore processes*, in the topic, *Disk Space Requirements* (Linux).

### Directory structure

The reference architecture recommends installing the Tableau Server package and the data into non-default locations:

- **Install package to:** `/app/tableau_server`: Create this directory path before you install the Tableau Server package, and then specify this path during installation.
- **Install Tableau data to:** `/data/tableau_data`. Do not create this directory before you install Tableau Server. Instead, you must specify the path during installation, and then Tableau Setup will create and permission the path appropriately.

See Run installation package and initialize TSM for implementation details.

### Example: Install and prep host computers in AWS

This section explains how to install EC2 hosts for each server type in the Tableau Server reference architecture.

The reference architecture requires eight hosts:

- Four instances for Tableau Server.
- Two instances for proxy servers (Apache).
- One instance for bastion host.
- One or two EC2 PostgreSQL database instances

### Host instance details

Install host computers according to the details below.
Tableau Server

- Amazon Linux 2
- Instance Type: r5.4xlarge
- Security group ID: Private
- Storage: EBS, 150 GiB, gp2 volume type. If your deployment will make use of external storage for the Tableau File Store, you will need calculate the appropriate disk space. See Install Tableau Server with External File Store (Linux).
- Network: install two EC2 hosts in each private subnet (10.0.30.0/24 and 10.0.31.0/24).
- Copy the latest maintenance release of Tableau Server 2021.2 (or later) rpm package from Tableau Downloads page to each Tableau host.

Bastion host

- Amazon Linux 2
- Instance Type: t3.micro
- Security group ID: Bastion
- Storage: EBS, 50 GiB, gp2 volume type
- Network: Bastion subnet 10.0.0.0/24

Proxy Server

- Amazon Linux 2
- Instance Type: t3.xlarge
- Security group ID: Public
- Storage: EBS, 100 GiB, gp2 volume type
- Network: Install one EC2 instance in each public subnet (10.0.1.0/24 and 10.0.2.0/24)

PostgreSQL EC2 host

- Amazon Linux 2
- Instance Type: r5.4xlarge
- Security group ID: Data
- Storage: Disk space requirement is dependent on your data load and how that will impact backup. See the section, Backup and restore processes, in the topic, Disk Space Requirements (Linux).
- Network: Data subnet 10.0.50.0/24. (If you are replicating PostgreSQL in a HA cluster, then install the second host in the 10.0.51.0/24 subnet)
Verification: VPC connectivity

After you have installed the host computers, verify network configuration. Verify connectivity between the hosts by connecting with SSH from the host in the Bastion security group to the hosts in each subnet.

Example: Connect to bastion host in AWS

1. Set up your admin computer for ssh-agent. This allows you to connect to hosts in AWS without placing your private key file on any EC2 instances.

   To configure ssh-agent on a Mac, run the following command:

   ```
   ssh-add -K myPrivateKey.pem or for latest Mac OS, ssh-add --apple-use-keychain myPrivateKey.pem
   ```

   For Windows, see the topic, Securely Connect to Linux Instances Running in a Private Amazon VPC.

2. Connect to the bastion host by running the following command:

   ```
   ssh -A ec2-user@<private-IP>
   ```

3. You can then connect to other hosts in the VPC from the bastion host, using the private IP address, for example:

   ```
   ssh -A ec2-user@10.0.1.93
   ```
Part 4 - Installing and Configuring Tableau Server

This topic describes how to finish installing and configuring the baseline Tableau Server deployment. The procedure here continues with the AWS and Linux reference architecture example.

The Linux examples throughout the installation procedures show commands for RHEL-like distributions. Specifically the commands here have been developed with the Amazon Linux 2 distribution. If you are running Ubuntu or Debian distributions, edit the commands accordingly.

Before you begin

You must prep and validate your environment as described in Part 3 - Preparing for Tableau Server Enterprise Deployment.
Install, configure, and tar PostgreSQL

This PostgreSQL instance hosts the external repository for the Tableau Server deployment. You must install and configure PostgreSQL before you install Tableau.

You can run PostgreSQL on Amazon RDS or on an EC2 instance. For more information about the differences between running the repository on RDS vs an EC2 instance, see Tableau Server External Repository (Linux).

By way of example, the procedure below shows how to install and configure Postgres on an Amazon EC2 instance. The example shown here is a generic installation and configuration for PostgreSQL in the reference architecture. Your DBA should optimize your PostgreSQL deployment based on the size of your data and performance needs.

Requirements: Note that you must be running PostgreSQL 1.6 and you must install the uuid-ossip module.

PostgreSQL versioning

You must install compatible major versions of PostgreSQL for the Tableau Server external repository. Additionally, minor versions must also meet minimum requirements:

<table>
<thead>
<tr>
<th>Tableau Server versions</th>
<th>PostgreSQL compatible versions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Major version</td>
</tr>
<tr>
<td>2021.2.3 - 2021.4</td>
<td>12</td>
</tr>
<tr>
<td>2022.1</td>
<td>13</td>
</tr>
<tr>
<td>2022.1.1</td>
<td>13</td>
</tr>
</tbody>
</table>

Install PostgreSQL

This example installation procedure describes how to install PostgreSQL version 13.4.

Sign-in to the EC2 host that you created in the previous Part.
1. Run update to apply latest fixes to the Linux OS:

   ```
sudo yum update
   ```

2. Create and edit the file, pgdg.repo, in the `/etc/yum.repos.d/` path. Populate the file with the following configuration information:

   ```
[pgdg13]
name=PostgreSQL 13 for RHEL/CentOS 7 - x86_64
baseurl = https://
//download.postgresql.org/pub/repos/yum/13/redhat/rhel-7-
x86_64
enabled=1
gpgcheck=0
```

3. Install Posgres 13.4:

   ```
sudo yum install postgresql13-server-13.4-1PGDG.rhel7.x86_64
   ```

4. Install the uuid-ossp module:

   ```
sudo yum install postgresql13-contrib-13.4-1PGDG.rhel7.x86_64
   ```

5. Initialize Postgres:

   ```
sudo /usr/pgsql-13/bin/postgresql-13-setup initdb
   ```

### Configure Postgres

Finish the base installation by configuring Postgres:

1. Update the `pg_hba` configuration file, `/var/lib/pgsql/13/data/pg_hba.conf`, with the following two entries. Each entry must include the mask of the subnets.
where your Tableau Servers will be running:

```bash
host all all 10.0.30.0/24 password
host all all 10.0.31.0/24 password
```

2. Update the PostgreSQL file, `/var/lib/pgsql/13/data/postgresql.conf`, by adding this line:

```bash
listen_addresses = '*'
```

3. Configure to start Postgres on reboot:

```bash
sudo systemctl enable --now postgresql-13
```

4. Set superuser password:

```bash
sudo su - postgres
psql -c "alter user postgres with password 'StrongPassword""
```

**Note:** Set a strong password. Do not use 'StrongPassword' as shown in the example here.

```bash
exit
```

5. Restart Postgres:

```bash
sudo systemctl restart postgresql-13
```

**Take PostgreSQL Step 1 tar backup**

Create a tar back up of the PostgreSQL configuration. Creating a tar snapshot of the current configuration will save you time if you encounter failures as you continue the deployment.
We'll refer to this as the "Step 1" backup.

On PostgreSQL host:

1. Stop the Postgres database instance:
   
   ```
   sudo systemctl stop postgresql-13
   ```

2. Run the following commands to create the tar backup:
   
   ```
   sudo su
   cd /var/lib/pgsql
   tar -cvf step1.13.bkp.tar
   exit
   ```

3. Start Postgres database:
   
   ```
   sudo systemctl start postgresql-13
   ```

---

**Restore Step 1**

Restore to Step 1 if the Tableau Server initial node fails during installation.

1. On the computer running Tableau, run the obliterate script to completely remove Tableau Server from the host:
   
   ```
   sudo /app/tableau_server/packages/scripts.<version_code>/./tableau-server-obliterate -a -y -y -y -l
   ```

2. Restore the PostgreSQL Stage 1 tar. On the computer running Postgres, run the following commands:
   
   ```
   sudo su
   systemctl stop postgresql-13
   cd /var/lib/pgsql
   ```
tar -xvf step1.13.bkp.tar

systemctl start postgresql-13

exit

3. Verify that the Tableau host can connect as a client to the Postgres database. Run the following command from the Tableau host computer:

```bash
psql postgresql://postgres@<IP_address>:5432/postgres
```

Resume the installation process of installing the initial node of Tableau Server.

**Before you install**

If you are deploying Tableau according to the example AWS/Linux implementation described in this Guide, then you may be able to run the automated installation script, EDGSetup. The EDGSetup script automates the example installation of the four-node Tableau deployment that is described in procedures that follow. See Appendix - AWS Deployment Toolbox.

**Install initial node of Tableau Server**

This procedure describes how install the initial node of Tableau Server as defined by the reference architecture. With the exception of the package installation and the initialization of TSM, the procedure here uses the TSM command line whenever possible. In addition to being platform-agnostic, using TSM CLI allows a more seamless installation into virtualized and headless environments.

**Run installation package and initialize TSM**

Sign in to the Node 1 host server.

1. Run update to apply latest fixes to the Linux OS:

```bash
sudo yum update
```
2. Copy the installation package from Tableau Downloads page to the host computer that will be running Tableau Server.

   For example, on a computer running Linux RHEL-like operating system, run

   ```bash
   wget https://-
downloads.tableau.com/esdalt/2022<version>/tableau-server-
<version>.rpm
   ```

   where `<version>` is the release number.

3. Download and install dependencies:

   ```bash
   sudo yum deplist tableau-server-<version>.rpm | awk '/provider:/ {print $2}' | sort -u | xargs sudo yum -y install
   ```

4. Create the `/app/tableau_server` path in the root directory:

   ```bash
   sudo mkdir -p /app/tableau_server
   ```

5. Run the installation program and specify the `/app/tableau_server` install path. For example, on a Linux RHEL-like operating system, run:

   ```bash
   sudo rpm -i --prefix /app/tableau_server tableau-server-
<version>.x86_64.rpm
   ```

6. Change to the `/app/tableau_server/packages/scripts.<version_code>/` directory and run the `initialize-tsm` script located there:

   ```bash
   sudo ./initialize-tsm -d /data/tableau_data --accepteula
   ```

7. After initialization is complete, exit the shell:

   ```bash
   exit
   ```
Activate and register Tableau Server

1. Sign in to the Node 1 host server.

2. Provide the Tableau Server Key and the IT Management Add-on key in this step.
   You will need to run the following command twice, first with the Tableau Server product key and then with the Server Management Add-on product key:

   ```
   tsm licenses activate -k <product key>
   ```

3. Create a json registration file with the format as shown here:

   ```
   {
   "zip": "97403",
   "country": "USA",
   "city": "Springfield",
   "last_name": "Simpson",
   "industry": "Energy",
   "eula": "yes",
   "title": "Safety Inspection Engineer",
   "company_employees": "100",
   "phone": "5558675309",
   "company": "Example",
   "state": "OR",
   "opt_in": "true",
   "department": "Engineering",
   "first_name": "Homer",
   "email": "homer@example.com"
   }
   ```

4. After saving changes to the file, pass it with the --file option to register Tableau Server:

   ```
   tsm register --file path_to_registration_file.json
   ```
Configure identity store

**Note:** If your deployment will make use of external storage for the Tableau File Store, you will need to enable External File Store before you configure the identity store. See *Install Tableau Server with External File Store (Linux)*.

The default reference architecture uses a local identity store. Configure the initial host with local identity store by passing the `config.json` file with the `tsm settings import` command.

Import the `config.json` file according to your operating system:

The `config.json` file is included in the `scripts.<version>` directory path (for example, `scripts.20204.21.0217.1203`), and is formatted to configure the identity store.

Run the following command to import the `config.json` file:

```
  tsm settings import -f /app/tableau_server-/packages/scripts.<version_code>/config.json
```

**Configure external Postgres**

1. Create an external database `json` file with the following configuration settings:

   ```json
   {
     "flavor":"generic",
     "masterUsername":"postgres",
     "host":"<instance ip address>",
     "port":5432
   }
   ```

2. After saving changes to the file, pass the file with the following command:

   ```
   tsm topology external-services repository enable -f
   ```
You will be prompted for the Postgres master username password.

3. Apply the changes.

Run this command to apply the changes and restart Tableau Server:

```shell
tsm pending-changes apply
```

4. Delete the configuration file that you used in Step 1.

Finish Node 1 installation

1. After Tableau Server has installed you must initialize the server.

Run the following command:

```shell
tsm initialize --start-server --request-timeout 1800
```

2. When initialization is finished, you must create a Tableau Server administrator account.

Unlike the computer account that you are using to install and manage TSM operating-system components, the Tableau Server administrator account is an application account that used for creating Tableau Server users, projects, and sites. The Tableau Server administrator also applies permissions to Tableau resources. Run the following command to create the initial administrator account. In the following example, the user is called `tableau-admin`:

```shell
tabcmd initialuser --server http://localhost --username "tableau-admin"
```

Tabcmd will prompt you to set a password for this user.
Verification: Node 1 configuration

1. Run the following command to verify that TSM services are running:

   tsm status -v

   Tableau should return the following:

   external:
   Status: RUNNING
   'Tableau Server Repository 0' is running (Active Repository).
   node1: localhost
   Status: RUNNING
   'Tableau Server Gateway 0' is running.
   'Tableau Server Application Server 0' is running.
   'Tableau Server Interactive Microservice Container 0' is running.
   'MessageBus Microservice 0' is running.
   'Relationship Query Microservice 0' is running.
   'Tableau Server VizQL Server 0' is running.
   ...

   All of the services will be listed.

2. Run the following command to verify that Tableau administrative site is running:

   curl localhost

   The first few lines should show Vizportal html, similar to this:

   <!DOCTYPE html>
   <html xmlns:ng="" xmlns:tb="">
   <head ng-csp>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="initial-scale=1, maximum-
Take Step 2 tar backups

After you have verified the initial installation, take two tar backups:

- PostgreSQL
- Tableau initial node (Node 1)

In most cases, you can recover your installation of the initial node by restoring these tar files. Restoring the tar files is much quicker than reinstalling and reinitializing the initial node.

Create Step 2 tar files

1. On the initial node of Tableau, stop Tableau:

   tsm stop

   Wait for Tableau to stop before continuing to the next step.

2. On PostgreSQL host, stop the Postgres database instance:

   sudo systemctl stop postgresql-13

3. Run the following commands to create the tar backup:

   sudo su
   cd /var/lib/pgsql
   tar -cvf step2.13.bkp.tar 13
   exit
4. Verify that the Postgres tar file is created with root permissions:

```bash
sudo ls -al /var/lib/pgsql
```

5. On the Tableau host, stop Tableau administrative services:

```bash
sudo /app/tableau_server/packages/scripts.<version_code>/./stop-administrative-services
```

6. Run the following commands to create the tar backup:

```bash
cd /data
sudo tar -cvf step2.tableau_data.bkp.tar tableau_data
```

7. On the Postgres host, start the Postgres database:

```bash
sudo systemctl start postgresql-13
```

8. On the Tableau host, verify that the Tableau host can connect and authenticate with PostgreSQL:

```bash
psql postgresql://postgres@<IP_address>:5432/postgres
```

9. Start Tableau administrative services:

```bash
sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services
```

10. Run the `tsm status` command to monitor TSM state before restarting.

    In most cases, the command will first return a status of DEGRADED or ERROR. Wait a few minutes and run the command again. If the status of ERROR or DEGRADED is returned, continue waiting. Do not attempt to start TSM until the status, STOPPED is returned. And then run the following command:

    ```bash
tsm start
    ```

**Restore Step 2**
This process restores the Tableau Node 1 and the Postgres instance to Step 2. After you have restored to this step, you can then redeploy the remaining Tableau Nodes.

1. Stop the tsm services on the initial Tableau host (Node 1):

   ```
   tsm stop
   ```

2. Stop Tableau administrative services on all nodes of the Tableau Server deployment. Run the following command on each node, in order (Node 1, Node 2, and then Node 3):

   ```
   sudo /app/tableau_server/packages/scripts.<version_code>/./stop-administrative-services
   ```

3. After Tableau services have stopped, restore the PostgreSQL Step 2 tar. On the computer running Postgres, run the following commands:

   ```
   sudo su
   systemctl stop postgresql-13
   cd /var/lib/pgsql
   tar -xvf step2.13.bkp.tar
   systemctl start postgresql-13
   exit
   ```

4. Verify that the Tableau host can connect as a client to the Postgres database. Run the following command from the Tableau host computer:

   ```
   psql postgresql://postgres@<IP_address>:5432/postgres
   ```

5. Restore the Tableau Step 2 tar. On the initial Tableau host, run the following commands:

   ```
   cd /data
   sudo rm -rf tableau_data
   ```
sudo tar -xvf step2.tableau_data.bkp.tar

6. On the Tableau Node 1 computer, remove the following files:
   - sudo rm /data/tableau_data/data/t-absvc/appzookeeper/0/version-2/currentEpoch
   - sudo rm /data/tableau_data/data/t-absvc/appzookeeper/0/version-2/acceptedEpoch
   - sudo rm /data/tableau_data/data/t-absvc/tabadminagent/0/servicestate.json

7. Start the Tableau administrative services:
   
   sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services

8. Reload the Tableau systemctl files and then run start-administrative-services again:

   sudo su -l tableau -c "systemctl --user daemon-reload"

   sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services

9. On Node 1, run the tsm status command to monitor TSM state before restarting.

   In some cases, you will get an error, Cannot connect to server.... This error occurs because the tabadmincontroller service has not restarted. Continue to run tsm status periodically. If this error does not go away after 10 minutes, run the start-administrative-services command again.

   After a few moments, the tsm status command will return a status of DEGRADED, and then ERROR. Do not start TSM until the status, STOPPED is returned. And then run the following command:
tsm start

Resume the installation process to install Tableau Server on remaining nodes.

Install Tableau Server on remaining nodes

To continue the deployment, copy the Tableau installer to each node.

Node configuration overview

This section describes the process to configure Nodes 2-4. The sections that follow provide detailed configuration and validation procedures for each step.

Installation of Tableau Server Nodes 2-4 requires that you generate, copy, and reference a bootstrap file during node installation.

To generate the bootstrap file, you run a TSM command on the initial node. You will then copy the bootstrap file to the target node, where you run it as part of the node initialization.

The following json content shows an example of a bootstrap file. (The certificate and crypto-related values have been truncated to make the example file easier to read.)

```json
{
  "initialBootstrapSettings" : {
    "certificate" : "-----BEGIN CERTIFICATE-----\r\n...\r
END CERTIFICATE-----",
    "port" : 8850,
    "configurationName" : "tabsvc",
    "clusterId" : "tabsvc-clusterid",
    "cryptoKeyStore" : "zs7OzgAAAAIAAAABAAAAA...w==",
    "toksCryptoKeystore" : "LS0tLS1CRUdJTiBUT00tLS0tCjM5MDBh...L",
    "sessionCookieMaxAge" : 7200,
    "nodeId" : "node1",
    "machineAddress" : "ip-10-0-1-93.us-west-1.compute.internal",
    "cryptoEnabled" : true,
    "sessionCookieUser" : "tsm-bootstrap-user",
  }
}``
The bootstrap file includes connection-based validation to authenticate Node 1 and creates an encrypted channel for the bootstrap process. The bootstrap session is time-limited, and configuring and validating nodes is time consuming. Plan on creating and copying new bootstraps as you configure the nodes.

After you run the bootstrap file, you then sign in to the initial Tableau Server node and configure the processes for the new node. When you finish configuring the nodes, you must apply changes and restart the initial node. The new node is configured and started. As you add nodes, the configuration and restart of the deployment will take consecutively longer to complete.

The Linux examples throughout the installation procedures show commands for RHEL-like distributions. If you are running Ubuntu or Debian distributions, edit the commands accordingly.

1. Run update to apply latest fixes to the Linux OS:
   
   sudo yum update

2. Download and install dependencies:

   sudo yum deplist tableau-server-<version>.rpm | awk '/provider:/ {print $2}' | sort -u | xargs sudo yum -y install

3. Create the /app/tableau_server path in the root directory:

   sudo mkdir -p /app/tableau_server

4. Run the installation program and specify the /app/tableau_server install path. For example, on a Linux RHEL-like operating system, run:
sudo rpm -i --prefix /app/tableau_server tableau-server-<version>.x86_64.rpm

Generate, copy, and use the bootstrap file to initialize TSM

The following procedure shows how to generate, copy, and use a bootstrap file when initializing TSM on another node. In this example, the bootstrap file is named boot.json.

In this example, the host computers are running in AWS, where EC2 hosts are running Amazon Linux 2.

1. Connect to the initial node (Node 1) and run the following command:

   tsm topology nodes get-bootstrap-file --file boot.json

2. Copy the bootstrap file to Node 2.

   scp boot.json ec2-user@10.0.31.83:/home/ec2-user/

3. Connect to Node 2 and switch to the Tableau Server scripts directory:

   cd /app/tableau_server/packages/scripts.<version_number>

4. Run the initialize-tsm command and reference the bootstrap file:

   sudo ./initialize-tsm -d /data/tableau_data -b /home/ec2-user/boot.json --accepteula

5. After initialize-tsm has completed, delete boot.json, and then exit or log out of the session.
Configure processes

You must configure the Tableau Server cluster on the node where the Tableau Server Administration Controller (TSM controller) is running. The TSM controller runs on the initial node.

### Process Status

The real-time status of processes running in Tableau Server.

<table>
<thead>
<tr>
<th>Process</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>External Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Controller</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Gateway</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Application Server</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VizQL Server</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Cache Server</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Search &amp; Browse</td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backgrounder</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Data Server</td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Engine</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>File Store</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Repository</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tableau Prep Conductor</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metrics</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Configure Node 2

1. After you have initialized TSM using the bootstrap file on Node 2, sign in to the initial node.

2. On the initial node (node1) run the following commands to configure processes on Node 2:
tsm topology set-process -n node2 -pr clustercontroller -c 1

3. Review the configuration before you apply it. Run the following command:

   tsm pending-changes list

4. After you have verified that your changes are in the pending list (there will be other services in the pending list as well), apply the changes:

   tsm pending-changes apply

   The changes will require a restart. Configuration and restart will take some time.

5. Verify Node 2 configuration. Run the following command:

   tsm status -v

**Configure Node 3**

Initialize TSM using the bootstrap process on Node 3, and then run the `tsm topology set-process` commands below.
There is a Coordination Service warning that will display each time you set a process. You can ignore this warning as you set the processes.

1. After you initialize TSM using the bootstrap file on Node 3, sign in to the initial node (node1) and run the following commands to configure processes:

   tsm topology set-process -n node3 -pr clustercontroller -c 1
   tsm topology set-process -n node3 -pr clientfileservice -c 1
   tsm topology set-process -n node3 -pr backgrounder -c 4
   tsm topology set-process -n node3 -pr filestore -c 1

   If you are installing version 2022.1 or later, add the Index and Search service as well:

   tsm topology set-process -n node3 -pr indexandsearchserver -c 1

2. Review the configuration before you apply it. Run the following command:

   tsm pending-changes list

3. After you have verified that your changes are in the pending list (the list will include other services that are automatically configured), apply the changes:

   tsm pending-changes apply --ignore-warnings

   The changes will require a restart. Configuration and restart will take some time.

4. Verify the configuration by running the following command:

   tsm status -v

**Deploy coordination service ensemble to Nodes 1-3**

For standard reference architecture four-node deployment, run the following procedure:
1. Run the following commands on Node 1:

```bash
tsm stop
tsm topology deploy-coordination-service -n node1,node2,node3
```

The process includes a restart of TSM, which will take some time.

2. After the coordination service is deployed, run `tsm start`.

Take Step 3 tar backups

After you have verified the installation, take four tar backups:

- PostgreSQL
- Tableau initial node (Node 1)
- Tableau Node 2
- Tableau Node 3

Create Step 3 tar files

1. On the initial node of Tableau, stop Tableau:

```bash
tsm stop
```

2. After TSM has stopped, stop Tableau administrative services on each node. Run the following command on each node, in order (Node 1, Node 2, and then Node 3):

```bash
sudo /app/tableau_server/packages/scripts.<version_code>/./stop-administrative-services
```

3. On PostgreSQL host, stop the Postgres database instance:

```bash
sudo systemctl stop postgresql-12
```

4. Run the following commands to create the tar backup:
sudo su

cd /var/lib/pgsql

tar -cvf step3.12.bkp.tar 12

exit

5. Verify that Postgres tar file is created with root permissions:

    sudo ls -al /var/lib/pgsql

6. On the Postgres host, start the Postgres database:

    sudo systemctl start postgresql-12

7. On the Tableau host, verify that the Tableau host can connect and authenticate with PostgreSQL:

    psql "postgresql://postgres@<IP address>:5432"/postgres

8. Create the tar backup on Node 1, Node 2, and Node 3. Run the following commands on each node:

    - cd /data
      
      sudo tar -cvf step3.tableau_data.bkp.tar tableau_data

    - Verify that the Tableau tar file is created with root permissions:
      
      ls -al

9. Start Tableau administrative services on each node in order (Node 1, Node 2, and then Node 3):

    sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services

10. Run the tsm status command to monitor TSM state before restarting.
In most cases, the command will return a status of DEGRADED, and then ERROR. Wait a few moments and run the command again. If the status of ERROR or DEGRADED is returned, continue waiting. Do not attempt to start TSM until the status, STOPPED is returned. And then run the following command:

```
tsm start
```

## Restore Step 3

This process restores the Tableau Node 1, Node 2, and Node 3. It also restores the PostgreSQL instance to Step 3. After you have restored to this step, you can then deploy coordination service, Node 4, and then final node configurations.

1. **Stop the tsm service on the initial Tableau host (Node 1):**

   ```
   tsm stop
   ```

2. **After TSM has stopped, stop Tableau administrative services on Node 1, Node 2, and Node 3. Run the following command on each node:**

   ```
sudo /app/tableau_server/packages/scripts.<version_code>/./stop-administrative-services
   ```

3. **Restore the PostgreSQL Step 3 tar. On the computer running Postgres, run the following commands:**

   ```
sudo su
systemctl stop postgresql-12
cd /var/lib/pgsql
tar -xvf step3.12.bkp.tar
systemctl start postgresql-12
```
4. Verify that the Tableau host can connect as a client to the Postgres database. Run the following command from the Tableau Node 1 host computer:

```sql
psql "postgresql://postgres@<IP-address>:5432/postgres?sslmode=verify-ca&sslrootcert=rootCACert.pem"
```

5. Restore the Tableau Step 3 tar on Node 1, Node 2, and Node 3. Run the following commands on each Tableau node:

```bash
cd /data
sudo rm -rf tableau_data
sudo tar -xvf step3.tableau_data.bkp.tar
```

6. On the Tableau Node 1 computer, remove the following files:

- `sudo rm /data/tableau_data/data/t-absvc/appzookeeper/1/version-2/currentEpoch`
- `sudo rm /data/tableau_data/data/t-absvc/appzookeeper/1/version-2/acceptedEpoch`
- `sudo rm /data/tableau_data/data/t-absvc/tabadminagent/0/servicestate.json`

If the shell returns a "file not found" error, you may need to change the path name to increment the number `<n>` in this section of the path: `.../app-zookeeper/<n>/version-2/...`

7. Restart the administrative services on Node 1, Node 2, and Node 3. Run the following commands on each node:

```bash
sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services
```
sudo su -l tableau -c "systemctl --user daemon-reload"

```
sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services
```

8. On Node 1, run the `tsm status` command to monitor TSM state before restarting.

   In some cases, you will get an error, Cannot connect to server.... This error occurs because the tabadmincontroller service has not restarted. Continue to run `tsm status` periodically. If this error does not go away after 10 minutes, run the `start-administrative-services` command again.

   After a few moments, the `tsm status` command will return a status of DEGRADED, and then ERROR. Do not start TSM until the STOPPED status is returned. And then run the following command:

   ```
   tsm start
   ```

Resume the installation process to deploy coordination service on Nodes 1-3.

**Configure Node 4**

The process for configuring Node 4 is the same as Node 3.

Set the same processes as you set for Node 3, running the same set of commands as shown above, but specifying `node4` in the commands rather than `node3`.

As with Node 3 verification, verify the Node 4 configuration by running `tsm status -v`.

Before you proceed, wait for the File Store process on Node 4 to finish synchronizing. The File Store service status will return is synchronizing until it finishes. When the File Store service status returns is running you can proceed.

**Final process configuration and verification**

The final step to process configuration is to remove redundant processes from Node 1.
1. Connect to the initial node (node1).

2. Decommission the file store on Node 1. This will cause a warning about removing the file store from a co-located controller. You can ignore the warning. Run the following command:

   tsm topology filestore decommission -n node1

3. When file store is decommissioned, run the following command to remove the backgrounder process from Node 1:

   tsm topology set-process -n node1 -pr backgrounder -c 0

4. Review the configuration before you apply it. Run the following command:

   tsm pending-changes list

5. After you have verified that your changes are in the pending list, apply the changes:

   tsm pending-changes apply

   The changes will require a restart. Configuration and restart will take some time.

6. Verify the configuration:

   tsm status -v.

   Before you proceed, wait for the File Store process on Node 4 to finish synchronizing. The File Store service status will return is synchronizing until it finishes. When the File Store service status returns is running you can proceed.

**Perform backup**

A full recovery of Tableau Server requires a backup portfolio that includes three components:

- A backup file of the repository and file store data. This file is generated by the `tsm maintenance backup` command.
A topology and configuration export file. This file is generated by the \texttt{tsm settings export} command.

- Authentication certificate, key, and keytab files.

For a full description of the backup and restore process, see the Tableau Server topic, \textit{Perform a Full Backup and Restore of Tableau Server (Linux)}.

At this stage of your deployment, all relevant files and assets that are required for a full restoration are included by running the \texttt{tsm maintenance backup} and \texttt{tsm settings export} commands.

1. Run the following command to export the configuration and topology settings to a file called \texttt{ts\_settings\_backup.json}

   \texttt{tsm settings export -f ts\_settings\_backup.json}

2. Run the following command to create a backup of the repository and file store data in a file named \texttt{ts\_backup-\<yyyy-mm-dd\>.tsbak}. Ignore the warning about the file store not being on the controller node.

   \texttt{tsm maintenance backup -f ts\_backup -d --skip-compression}

   Location of backup file:

   \texttt{/data/tableau\_data/data/tabsvc/files/backups/}

3. Copy both files and save them on a different storage asset that is not shared by your Tableau Server deployment.
Part 5 - Configuring Web Tier

The web tier of the reference architecture should include the following components:

- A web-facing application load balancer that accepts HTTPS requests from Tableau clients and communicates with the reverse proxy servers.
- Reverse proxy:
  - We recommend a minimum of two proxy servers for redundancy and to handle client load.
  - Receives HTTPS traffic from load balancer.
  - Supports sticky session to Tableau host.
  - Configure proxy for round robin load balancing to each Tableau Server running the Gateway process.
  - Handles authentication requests from external IdP.
- Forward proxy: Tableau Server requires access to the internet for licensing and map functionality. You must configure forward proxy safelists for Tableau service URLs. See *Communicating with the Internet (Linux)*.
- All client-related traffic may be encrypted over HTTPS:
  - Client to application load balancer
  - Application load balancer to reverse proxy servers
  - Proxy server to Tableau Server
  - Authentication handler running on reverse proxy to IdP
  - Tableau Server to IdP
**Note:** Tableau Server version 2022.1 includes the Tableau Server Independent Gateway. The Independent Gateway is a standalone instance of the Tableau Gateway process that serves as a Tableau-aware reverse proxy. At the time of release, the Independent Gateway has been validated, but not fully tested in the EDG reference architecture. After full testing is complete the EDG will be updated with Tableau Server Independent Gateway prescriptive guidance.

## Authentication and authorization

The default reference architecture specifies installing Tableau Server with local authentication configured. In this model, clients must connect to Tableau Server to be authenticated by the native Tableau Server local authentication process. We do not recommend using this authentication method in the reference architecture because the scenario requires that unauthenticated clients communicate into the application tier, which is a security risk.

Instead, we recommend configuring an enterprise grade external identity provider coupled with an AuthN module to pre-authenticate all traffic to the application tier. When configured with an external IdP, the native Tableau Server local authentication process is not used. Tableau Server authorizes access to resources in the deployment after the IdP has authenticated the users.

### Pre-authentication with an AuthN module

In the example documented in this Guide, SAML SSO is configured, but the pre-authentication process can be configured with most external identity providers and an AuthN module.

In the reference architecture the reverse proxy is configured to create a client authentication session with the IdP before proxying those request to Tableau Server. We refer to this process as the *pre-auth* phase. The reverse proxy will only redirect authenticated client
Tableau Server will then create a session, verify authentication of the session with the IdP, and then return the client request.

The following diagram shows the step-by-step detail of the pre-auth and authentication process with an AuthN module configured:

Configuration overview

This is an overview of the process to configure the web tier. Verify connectivity after each step:

1. Configure two reverse proxies to provide HTTP access to Tableau Server.
2. Configure load balancing logic with sticky sessions on proxy servers to connect to each Tableau Server instance running the Gateway process.
3. Configure application load balancing with sticky sessions at the internet gateway to forward requests to the reverse proxy servers.

4. Configure authentication with an external IdP. You can configure SSO or SAML by installing an authentication handler on the reverse proxy servers. The AuthN module manages the authentication handshake between the external IdP and your Tableau deployment. Tableau will also act as an IdP service provider and authenticate users with the IdP.

5. To authenticate with Tableau Desktop in this deployment, your clients must be running Tableau Desktop 2021.2.1 or later.

6. Optional: If your organization is running Tableau Desktop 2021.4 or later, then you must run `tsm configuration set -k features.ExternalBrowserOAuth -v true` on Tableau Server to enable authentication through the reverse proxy servers.

Example web tier configuration

The remainder of this topic provides an end-to-end procedure that describes how to implement web tier in the example AWS reference architecture. The example configuration is composed of the following components:

- AWS application load balancer
- Apache proxy servers
- Mellon authentication module
- Okta IdP
- SAML authentication

**Note:** The example web tier configuration presented in this section includes detailed procedures for deploying third party software and services. We’ve made a best effort to verify and document the procedures to enable the web tier scenario. However, the third-party software may change or your scenario may differ from the reference architecture described here. Please refer to third-party documentation for authoritative configuration details and support.
The Linux examples throughout this section show commands for RHEL-like distributions. Specifically the commands here have been developed with the Amazon Linux 2 distribution. If you are running Ubuntu or Debian distributions, edit the commands accordingly.

Deploying the web tier in this example follows a stepwise configuration and verification procedure. The core web tier configuration consists of the following steps to enable HTTP between Tableau and the internet. Apache is run and configured for reverse proxy/load balancing behind the AWS application load balancer:

1. Install Apache
2. Configure reverse proxy to test connectivity to Tableau Server
3. Configure load balancing on proxy
4. Configure AWS application load balancer

After the web tier is set up and connectivity with Tableau is verified, configure authentication with an external provider.

Install Apache

Run the following procedure on both EC2 hosts (Proxy 1 and Proxy 2). If you are deploying in AWS according to the reference architecture example then you should have two availability zones and be running a single proxy server in each zone.

1. Install Apache:

   ```bash
   sudo yum update -y
   sudo yum install -y httpd
   ```

2. Configure to start Apache on reboot:

   ```bash
   sudo systemctl enable --now httpd
   ```

3. Verify that the version of httpd you have installed includes `proxy_hcheck_module`:

   ```bash
   sudo httpd -M
   ```
The `proxy_hcheck_module` is required. If your version of httpd does not include this module, then update to a version of httpd that does include it.

**Configure proxy to test connectivity to Tableau Server**

Run this procedure on one of the proxy hosts (Proxy 1). The purpose of this step is to verify connectivity between the internet to your proxy server to the Tableau Server in the private security group.

1. Create a file called `tableau.conf` and add it to the `/etc/httpd/conf.d` directory.

   Copy the following code and specify the `ProxyPass` and `ProxyPassReverse` keys with the private IP address of Tableau Server Node 1.

   ```xml
   <VirtualHost *:80>
   ProxyPreserveHost On
   ProxyPass "/" "http://10.0.30.32:80/
   ProxyPassReverse "/" "http://10.0.30.32:80/
   </VirtualHost>
   ```

   **Important:** The configuration shown below is not a secure and should not be used in production. This configuration should only be used during the installation process to verify end-to-end connectivity.

2. Restart httpd:

   ```bash
   sudo systemctl restart httpd
   ```
Verification: Base topology configuration

You should be able to access the Tableau Server admin page by browsing to http://<proxy-public-IP-address>.

If the Tableau Server sign-in page does not load in your browser then follow these troubleshooting steps on the Proxy 1 host:

- Stop and then start httpd as a first troubleshooting step.
- Double-check the tableau.conf file. Verify that the Node 1 private IP is correct. Verify double-quotes and carefully review syntax.
- Run the curl command on the reverse proxy server with Node 1 private IP address, for example, curl 10.0.1.90. If the shell does not return html, or if it returns html for the Apache test web page, then verify protocol/port configuration between the Public and Private security groups.
- Run the curl command with Proxy 1 private IP address, for example, curl 10.0.0.163. If the shell returns the html code for the Apache test web page, then the proxy file is not configured correctly.
- Always restart httpd (sudo systemctl restart httpd) after any configuration change to the proxy file or to the security groups.
- Make sure TSM is running on Node 1.

Configure load balancing on proxy

1. On the same proxy host (Proxy 1) where you created the tableau.conf file, remove the existing Virtual Host configuration and edit the file to include load-balancing logic.

   For example:

   ```
   <VirtualHost *:80>
   ServerAdmin admin@example.com
   #Load balancing logic.
   ProxyHCExpr ok234 {REQUEST_STATUS} =~ /[^234]/
   Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e;
p@th="/" env=BALANCER_ROUTE_CHANGED
   ```
<Proxy balancer://tableau>
#Replace IP addresses below with the IP addresses to the 
Tableau Servers running the Gateway service.
BalancerMember http://10.0.3.40/ route=1 hcmethod=GET hcex-
pr=ok234 hcuri=/favicon.ico
BalancerMember http://10.0.4.151/ route=2 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPreserveHost On
ProxyPass / balancer://tableau/
ProxyPassReverse / balancer://tableau/
</VirtualHost>

2. Stop and then start httpd:

   sudo systemctl stop httpd
   sudo systemctl start httpd

3. Verify the configuration by browsing to the public IP address of Proxy 1.

Copy configuration to second proxy server

1. Copy the `tableau.conf` file from Proxy 1 and save it to the `/etc/httpd/conf.d` directory on Proxy 2 host.

2. Stop and then start httpd:

   sudo systemctl stop httpd
   sudo systemctl start httpd

3. Verify the configuration by browsing to the public IP address of Proxy 2.

Configure AWS application load balancer

Configure the load balancer as an HTTP listener. The procedure here describes how to add a load balancer in AWS.
Step 1: Create target group

A target group is an AWS configuration that defines the EC2 instances running your proxy servers. These are the targets for traffic from the LBS.

1. **EC2 > Target groups > Create target group**

2. On Create page:
   - Enter a target group name, for example `TG-internal-HTTP`
   - Target type: Instances
   - Protocol: HTTP
   - Port: 80
   - VPC: Select your VPC
   - Under **Health checks > Advanced health checks settings > Success codes**, append the code list to read: `200, 303`.
   - Click **Create**

3. Select the target group that you just created, and then click the **Targets** tab:
   - Click **Edit**.
   - Select the EC2 instances that are running proxy application, and then click **Add to registered**.
   - Click **Save**.

Step 2: Launch load balancer wizard

1. **EC2 > Load Balancers > Create Load Balancer**

2. On "Select load balancer type" page, create an Application Load Balancer.

**Note:** The UI that is displayed to configure load balancer is not consistent across AWS datacenters. The procedure below, "Wizard configuration," maps to the AWS configuration wizard that begins with **Step 1 Configure Load Balancer**.

If your datacenter displays all configurations in a single page that includes a **Create**
load balancer button at the bottom of the page, then follow the "Single page configuration" procedure below.

Wizard configuration

1. **Configure load balancer** page:
   - Specify name
   - Scheme: internet-facing (default)
   - IP address type: ipv4 (default)
   - Listeners (Listeners and routing):
     a. Leave the default HTTP listener
     b. Click **Add listener** and add HTTPs:443
   - VPC: select the VPC where you've installed everything
   - Availability Zones:
     - Select the a and b for your datacenter regions
     - In each corresponding drop-down selector, select the Public subnet (where your proxy servers reside).
     - Click: **Configure Security Settings**

2. **Configure Security Settings** page
   - Upload your public SSL certificate.
   - Click **Next: Configure Security Groups**.

3. **Configure Security Groups** page:
   - Select the Public security group. If the Default security group is selected, then clear that selection.
   - Click **Next: Configure Routing**.

4. **Configure Routing** page
   - Target group: Existing target group.
   - Name: Select target group that you created earlier
   - Click **Next: Register Targets**.

5. **Register Targets** page
The two proxy server instances that you configured previously should be displayed.
Click **Next: Review**.

6. **Review** page

Click **Create**.

**Single page configuration**

**Basic configuration**

- Specify name
- Scheme: internet-facing (default)
- IP address type: ipv4 (default)

**Network mapping**

- VPC: select the VPC where you've installed everything
- Mappings:
  - Select the a and b (or comparable) Availability Zones for your datacenter regions
  - In each corresponding drop-down selector, select the Public subnet (where your proxy servers reside).

**Security groups**

Select the Public security group. If the Default security group is selected, then clear that selection.

**Listeners and routing**

- Leave the default HTTP listener. For **Default action**, specify the Target Group that you previously set up.
- Click **Add listener** and add **HTTPS:443**. For **Default action**, specify the Target Group that you previously set up.

**Secure listener settings**
- Upload your public SSL certificate.

Click **Create load balancer**.

**Step 3: Enable stickiness**

1. After the load balancer is created, you must enable stickiness on the Target Group.
   - Open AWS Target Group page (EC2 > Load Balancing > Target groups), select the target group instance that you just set up. On the Action menu, select **Edit attributes**.
   - On the **Edit attributes** page, select **Stickiness**, specify a duration of 1 day, and then **Save changes**.

2. On load balancer, enable stickiness on the HTTP listener. Select the load balancer you just configured, and then click the **Listeners** tab:
   - For **HTTP:80**, click **View/edit rules**. On the resulting **Rules** page, click the edit icon (once at the top of the page, and then again by the rule) to edit the rule. Delete the existing THEN rule and replace it by clicking **Add action > Forward to**. In the resulting THEN configuration, specify the same target group you have created. Under Group-level stickiness, enable stickiness and set duration to 1 day. Save the setting and then click **Update**.

**Step 4: Verify LBS connectivity**

Open AWS Load Balancer page (EC2 > Load Balancers), select the load balancer instance that you just set up.

Under **Description**, copy the DNS name and paste it into a browser to access the Tableau Server sign in page.

If you get a 500-level error, then you may need to restart your proxy servers.

**Update DNS with public Tableau URL**

Use your domain DNS zone name from the AWS Load Balancer description to create a CNAME value in your DNS. Traffic to your URL (tableau.example.com) should be sent to
the AWS public DNS name.

Verify connectivity

After your DNS updates are finished, you should be able to navigate to the Tableau Server sign-in page by entering your public URL, for example, https://tableau.example.com.

Example authentication configuration: SAML with external IdP

The following example describes how to setup and configure SAML with Okta IdP and Mellon authentication module for a Tableau deployment running in the AWS reference architecture. The example describes how to configure Tableau Server and the Apache proxy servers to use HTTP. Okta will send request to the AWS load balancer over HTTPS, but all internal traffic will travel over HTTP. As you configure for this scenario, be aware of the HTTP vs HTTPS protocols when setting URL strings.

This example uses Mellon as a pre-authentication service provider module on the reverse proxy servers. This configuration ensures that only authenticated traffic connects to Tableau Server, which also acts as a service provider with the Okta IdP. Therefore, you must configure two IdP applications: one for the Mellon service provider and one for the Tableau service provider.

Create Tableau administrator account

A common mistake when configuring SAML is to forget to create an administrator account on Tableau Server before enabling SSO.

The first step is to create an account on Tableau Server with a Server Administrator role. For the example Okta scenario, the username must be in a valid email address format, for
example, user@example.com. You must set a password for this user, but the password will not be used after SAML is configured.

Configure Okta pre-auth application

The end-to-end scenario described in this section requires two Okta applications:

- Okta pre-auth application
- Okta Tableau Server application

Each of these applications are associated with different metadata that you will need to configure on the reverse proxy and Tableau Server, respectively.

This procedure describes how to create and configure the Okta pre-auth application. Later in this topic you will create the Okta Tableau Server application. For a free test Okta account with limited users, see the Okta Developer web page.

Create a SAML app integration for the Mellon pre-authentication service provider.

1. Open the Okta administration dashboard > Applications > Create App Integration.

2. On Create a new app integration page, select SAML 2.0 and then click Next.

3. On the General Settings tab, enter an App name, for example Tableau Pre-Auth, and then click Next.

4. On the Configure SAML tab:

   - Single sign on URL. The final element of the path in the single sign on URL is referred to as the MellonEndpointPath in the mellon.conf configuration file that follows later in this procedure. You can specify whatever endpoint you would like. In this example, sso is the endpoint. The last element, postResponse, is required: https://tableau.example.com/sso/postResponse.
   - Clear the checkbox: Use this for Recipient URL and Destination URL.
Recipient URL: Same as SSO URL, but with HTTP. For example, http://tableau.example.com/sso/postResponse.
Destination URL: same as Recipient URL, but with HTTP. For example, http://tableau.example.com/sso/postResponse.
Name ID format: EmailAddress
Application username: Email
Attributes Statements: Name = mail; Name format = Unspecified; Value = user.email.

Click Next.

5. On the Feedback tab, select:
   - I'm an Okta customer adding an internal app
   - This is an internal app that we have created
   - Click Finish.

6. On the application Settings page, click the Identity Provider metadata link to launch a browser. Copy the browser link. This is the link you will use when you configure Mel- lon in the procedure that follows.

7. (Optional) Configure multifactor authentication:
   - In Okta: Applications > Applications > Your new application (e.g., Tableau Pre-Auth) > Sign On
   - Under Sign On Policy, click Add Rule.
   - On the App Sign On Rule, specify a name and the different MFA options. To test functionality, you can leave all options as default. However, under Actions, you must select, Prompt for factor, and then specify how often users must sign in. Click Save.

Create and assign Okta user

1. In Okta, create a user with the same username that you created in Tableau (user@example.com): Directory > People > Add person.
2. After the user is created, assign the new Okta app to that person: Click the user name then assign the application in Assign Application.

Install Mellon for pre-auth

1. On the EC2 instances that are running the Apache proxy server, run the following commands to install PHP and Mellon modules:

   
sudo yum install httpd php mod_auth_mellon

2. Create the /etc/httpd/mellon directory

Configure Mellon as pre-auth module

Run this procedure on both proxy servers.

1. Change directory:

   cd /etc/httpd/mellon

2. Create the service provider metadata. Run the mellon_create_metadata.sh script. You must include the entity ID and the return URL for your organization in the command.

   The return URL is referred to as the single sign on URL in Okta. The final element of the path in the return URL is referred to as the MellonEndpointPath in the mel-lon.conf configuration file that follows later in this procedure. In this example, we specify sso as the endpoint path.

   For example:

   
sudo /usr/libexec/mod_auth_mellon/mellon_create_metadata.sh https://tableau.example.com "https://t-ableau.example.com/sso"

   The script returns the service provider certificate, key, and metadata files.
3. Rename the service provider files in the `mellon` directory for easier readability. We will refer to these files by the following names in the documentation:

```
sudo mv *.key mellon.key
sudo mv *.cert mellon.cert
sudo mv *.xml sp_metadata.xml
```

4. From the same directory, download the Okta pre-auth application metadata. Rename the file to `idp_metadata.xml` when you download it. For example:

```
sudo wget https://example.okta.com/app/exk163uey5h0wG0pw5d7/sso/saml/metadata -O idp_metadata.xml
```

5. Create the `mellon.conf` file in the `/etc/httpd/conf.d` directory:

```
sudo nano /etc/httpd/conf.d/mellon.conf
```

6. Copy the following contents into `mellon.conf`.

```
<Location />
MellonSPPrivateKeyFile /etc/httpd/mellon/mellon.key
MellonSPCertFile /etc/httpd/mellon/mellon.cert
MellonSPMetadataFile /etc/httpd/mellon/sp_metadata.xml
MellonIdPMetadataFile /etc/httpd/mellon/idp_metadata.xml
MellonEndpointPath /sso
MellonEnable "info"
</Location>
```

7. Add the following contents into the existing `tableau.conf` file:

```
Inside the `<VirtualHost *:80>` block, add the following content. Update `ServerName` with the public host name in your Entity ID:

DocumentRoot /var/www/html
ServerName tableau.example.com
```
ServerSignature Off
ErrorLog logs/error_sp.log
CustomLog logs/access_sp.log combined
LogLevel info

Add the Location block outside of the <VirtualHost *:80> block. Update MellonCookieDomain with the top-level domain to preserve cookie information as shown:

<Location />
AuthType Mellon
MellonEnable auth
Require valid-user
MellonCookieDomain example.com
</Location>

The complete tableau.conf file should look like the following example:

<VirtualHost *:80>
ServerAdmin admin@example.com
ProxyHCExpr ok234 %{REQUEST_STATUS} =~ /^[234]$/
Header add Set-Cookie "ROUTEID=%{BALANCER_WORKER_ROUTE}e;
path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://tableau>
BalancerMember http://10.0.3.36/ route=1 hcmethod=GET hcex-
pr=ok234 hcuri=/favicon.ico
BalancerMember http://10.0.4.15/ route=2 hcmethod=GET hcex-
pr=ok234 hcuri=/favicon.ico
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPreserveHost On
ProxyPass / balancer://tableau/
ProxyPassReverse / balancer://tableau/
DocumentRoot /var/www/html
ServerName tableau.example.com
ServerSignature Off
ErrorLog logs/error_sp.log
CustomLog logs/access_sp.log combined
LogLevel info
</VirtualHost>
<Location />
AuthType Mellon
MellonEnable auth
Require valid-user
MellonCookieDomain example.com
</Location>

8. Verify the configuration. Run the following command:

```
sudo apachectl configtest
```

If the configuration test returns an error, fix any errors and run configtest again. A successful configuration will return, Syntax OK.

9. Restart httpd:

```
sudo systemctl restart httpd
```

Create Tableau Server application in Okta

1. In Okta dashboard: Applications > Applications > Browse App Catalog
2. In Browse App Integration Catalog, search Tableau, select the Tableau Server tile, and then click Add.
3. On Add Tableau Server > General Settings, enter a Label, and then click Next.
4. In Sign-On Options, select SAML 2.0, and then scroll down to Advanced Sign-on Settings:
   - **SAML Entity ID**: enter the public URL, for example, https://tableau.example.com.
   - **Application user name format**: Email
5. Click the Identity Provider metadata link, to launch a browser. Copy the browser link. This is the link you will use when you configure Tableau in the procedure that follows.

6. Click Done.

7. Assign the new Tableau Server Okta app to your user (user@example.com): Click the user name then assign the application in Assign Application.

Enable SAML on Tableau Server for IdP

Run this procedure on Tableau Server Node 1.

1. Download the Tableau Server application metadata from Okta. Use the link that you saved from the previous procedure:

   wget https://dev-66144217.okta-com/app/exklegxgt1fhjkSeS5d7/sso/saml/metadata -O idp_metadata.xml

2. Copy a TLS certificate and related key file to the Tableau Server. The key file must be an RSA key. For more information about SAML certificate and IdP requirements, see SAML Requirements (Linux).

   To simplify certificate management and deployment, and as a security best practice, we recommend using certificates generated by a major trusted-third party certificate authority (CA). Alternatively, you may generate self-signed certificates or use certificates from a PKI for TLS.

   If you do not have a TLS certificate, you can generate a self-signed certificate using the embedded procedure below.

Generate a self-signed certificate

Run this procedure on Tableau Server Node 1.
Tableau Server Enterprise Deployment Guide

a. Generate signing root certificate authority (CA) key:

```bash
openssl genrsa -out rootCAKey-saml.pem 2048
```

device.zheng@163.com

d. Sign the new certificate with the CA certificate that you created above. The following command also outputs the certificate in the **crt** format:

```bash
openssl x509 -req -in server-saml.csr -days 3650 -CA rootCACert-saml.pem -CAkey rootCAKey-saml.pem -CAcreateserial -out server-saml.crt
```

e. Convert the key file to RSA. Tableau requires an RSA key file for SAML. To

```bash
openssl rsa -in rootCAKey-saml.pem -out rootCAKey-saml.pem
```
convert the key, run the following command:

openssl rsa -in server-saml.key -out server-saml-rsa.key

3. Configure SAML. Run the following command, specifying your entity ID and return URL, and the paths to the metadata file, certificate file, and key file:

```bash
tsm authentication saml configure --idp-entity-id "https://tableau.example.com" --idp-return-url "https://tableau.example.com" --idp-metadata idp_metadata.xml --cert-file "server-saml.crt" --key-file "server-saml-rsa.key"

tsm authentication saml enable
```

4. If your organization is running Tableau Desktop 2021.4 or later, then you must run the following command to enable authentication through the reverse proxy servers.

Versions of Tableau Desktop 2021.2.1 - 2021.3 will work without running this command, provided that your pre-authentication module (e.g., Mellon) is configured to allow top-level domain cookie preservation.

```bash
tsm configuration set -k features.ExternalBrowserOAuth -v false
```

5. Apply configuration changes:

```bash
tsm pending-changes apply
```

**Validate SAML functionality**

To validate end-to-end SAML functionality, sign-in to Tableau Server with the public URL (e.g., https://tableau.example.com) with the Tableau admin account that you created at the beginning of this procedure.
Validation troubleshooting

**Bad Request:** A common error for this scenario is a "Bad Request" error from Okta. Often this issue occurs when the browser is caching data from previous Okta session. For example, if you manage the Okta applications as an Okta administrator and then attempt to access Tableau using a different Okta-enabled account, session data from the administrator data may cause the "Bad Request" error. If this error persists even after you clear the local browser cache, try validating the Tableau scenario by connecting with a different browser.

**Not Found - The requested URL was not found on this server:** This error indicates one of many configuration errors.

If the user is authenticated with Okta, and then receives this error, then it's likely that you have uploaded the Okta pre-auth application to Tableau Server when you configured SAML. Verify that you have the Okta Tableau Server application metadata configured on Tableau Server, and not the Okta pre-auth application metadata.

Other troubleshooting steps:

- Review `tableau.conf` carefully for typos or configuration errors
- Review the Okta pre-auth application settings. Be sure HTTP vs HTTPS protocols are set as specified in this topic.
- Restart `httpd` on both proxy servers.
- Verify that `sudo apachectl configtest` returns "Syntax OK" on both proxy servers.
- Verify that the test user is assigned to both applications in Okta.
- Verify that stickiness is set on the load balancer and associated target groups.
Part 6 - Post-Installation Configuration

Configure SSL/TLS from load balancer to Tableau Server

Some organizations require end-to-end encryption channel from the client to the back end service. The default reference architecture as described to this point specifies SSL from the client to the load balancer running in the web tier of your organization.

To configure SSL from the load balancer to Tableau Server, you must:

- Install a valid SSL certificate on both the Tableau and proxy servers.
- Configure SSL from the load balancer to the reverse proxy servers.
- Configure SSL from the proxy servers to Tableau Server.
- You may also configure SSL from Tableau Server to the PostgreSQL instance.

The remainder of this topic describes this implementation in the context of the example AWS example reference architecture.

Example: Configure SSL/TLS in AWS reference architecture

This section describes how to generate a self-signed certificate using OpenSSL, configure SSL on Tableau, and configure SSL on an Apache proxy server, all running in the example AWS reference architecture.

The Linux procedures throughout this example show commands for RHEL-like distributions. Specifically the commands here have been developed with the Amazon Linux 2
distribution. If you are running Ubuntu or Debian distributions, edit the commands accordingly.

To simplify certificate management and deployment, and as a security best practice, we recommend using certificates generated by a major trusted-third party certificate authority (CA). Alternatively, you may generate self-signed certificates or use certificates from a PKI for TLS.

**Step 1: Generate a self-signed certificate**

Run this procedure on one of the proxy hosts. After you generate the certificate and associated key, you will share it to the other proxy host and to Tableau Server Node 1.

1. Generate signing root certificate authority (CA) key:

   ```bash
   openssl genrsa -out rootCAKey.pem 2048
   ```

2. Create the root CA certificate:

   ```bash
   openssl req -x509 -sha256 -new -nodes -key rootCAKey.pem -days 3650 -out rootCACert.pem
   ```

   You will be prompted to enter values for the certificate fields. For example:

   - **Country Name (2 letter code) [XX]:** US
   - **State or Province Name (full name) []:** Washington
   - **Locality Name (eg, city) [Default City]:** Seattle
   - **Organization Name (eg, company) [Default Company Ltd]:** Tableau
   - **Organizational Unit Name (eg, section) []:** Operations
   - **Common Name (eg, your name or your server's hostname) []:** tableau.example.com
   - **Email Address []:** example@tableau.com

3. Create the certificate and related key (`serverssl.csr` and `serverssl.key` in the...
example below). The subject name for the certificate must match the public host name of the Tableau host. The subject name is set with the -subj option with the format "/CN=<host-name>", for example:

```bash
openssl req -new -nodes -text -out serverssl.csr -keyout serverssl.key -subj "/CN=tableau.example.com"
```

4. Sign the new certificate with the CA certificate that you created in step 2. The following command also outputs the certificate in the crt format:

```bash
openssl x509 -req -in serverssl.csr -days 3650 -CA rootCACert.pem -CAkey rootCAKey.pem -CAcreateserial -out serverssl.crt
```

### Step 2: Configure proxy server for SSL

Run this procedure on both proxy servers.

1. Install the Apache ssl module:

   ```bash
   sudo yum install mod_ssl
   ```

2. Create the /etc/ssl/private directory:

   ```bash
   sudo mkdir -p /etc/ssl/private
   ```

3. Copy the crt and key files to the following /etc/ssl/ paths:

   ```bash
   sudo cp serverssl.crt /etc/ssl/certs/
   sudo cp serverssl.key /etc/ssl/private/
   ```

4. Update the existing `tableau.conf` with the following updates:
Tableau Server Enterprise Deployment Guide

- Add the SSL rewrite block:
  
  ```
  RewriteEngine on
  RewriteCond %{SERVER_NAME} =tableau.example.com
  RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
  ```

- In the SSL rewrite block, update the RewriteCond server name: add your public host name, for example, `tableau.example.com`.
- Change `<VirtualHost *:80>` to `<VirtualHost *:443>`.
- Wrap the `<VirtualHost *:443>` and the `<Location />` blocks with `<IfModule mod_ssl.c> ... </IfModule>`.
- BalancerMember: change the protocol from `http` to `https`.
- Add SSL* elements inside the `<VirtualHost *:443>` block:

  ```
  SSLEngine on
  SSLCertificateFile /etc/ssl/certs/serverssl.crt
  SSLCertificateKeyFile /etc/ssl/private/serverssl.key
  SSLProxyEngine on
  SSLProxyVerify none
  SSLProxyCheckPeerName off
  SSLProxyCheckPeerExpire off
  ```

- In the LogLevel element: add `ssl:warn`.
- Optional: If you have installed and configured an authentication module, then you may have additional elements in the `tableau.conf` file. For example, the `<Location />` block will include elements.

An example `tableau.conf` file configured for SSL is shown here:

```
RewriteEngine on
RewriteCond %{SERVER_NAME} =tableau.example.com
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]

<IfModule mod_ssl.c>
<VirtualHost *:443>
```
ServerAdmin admin@example.com
ProxyHCExpr ok234 {REQUEST_STATUS} =~ /^[234]/
Header add Set-Cookie "ROUTEID={BALANCER_WORKER_ROUTE}e;
path="/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://tableau>
BalancerMember https://10.0.3.36/ route=1 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
BalancerMember https://10.0.4.15/ route=2 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPreserveHost On
ProxyPass / balancer://tableau/
ProxyPassReverse / balancer://tableau/
DocumentRoot /var/www/html
ServerName tableau.example.com
ServerSignature Off
ErrorLog logs/error_sp.log
CustomLog logs/access_sp.log combined
LogLevel info ssl:warn
SSLEngine on
SSLCertificateFile /etc/ssl/certs/serverssl.crt
SSLCertificateKeyFile /etc/ssl/private/serverssl.key
SSLProxyEngine on
SSLProxyVerify none
SSLProxyCheckPeerName off
SSLProxyCheckPeerExpire off
</VirtualHost>
<Location />
#If you have configured a pre-auth module (e.g. Mellon)
include those elements here.
</Location>
</IfModule>
Tableau Server Enterprise Deployment Guide

5. Add index.html file to suppress 403 errors:

```bash
sudo touch /var/www/html/index.html
```

6. Restart httpd:

```bash
sudo systemctl restart httpd
```

---

Step 3: Configure Tableau Server for external SSL

Copy the serverssl.crt and serverssl.key files from the Proxy 1 host to the initial Tableau Server (Node 1).

Run the following commands on Node 1:

```bash
tsm security external-ssl enable --cert-file serverssl.crt --key-file serverssl.key
tsm pending-changes apply
```

---

Step 4: Optional authentication configuration

If you have configured an external identity provider for Tableau, then you will likely need to update return URLs in the IdP administrative dashboard.

For example, if you are using a Okta pre-auth application, you will need to update the application to use HTTPS protocol for the Recipient URL and the Destination URL.

---

Step 5: Configure AWS load balancer for HTTPS

If you are deploying with AWS load balancer as documented in this guide, then you reconfigure the AWS load balancer to send HTTPS traffic to the proxy servers:

1. Deregister existing HTTP target group:

   In Target Groups, select the HTTP target group that has been configured for the load balancer, click Actions, and then click Register and deregister instance.
On the Register and deregister targets page, select the instances that are currently configured, click Deregister, and then click Save.

2. Create HTTPS target group:

Target groups > Create target group

- Select "Instances"
- Enter a target group name, for example TG-internal-HTTPS
- Select your VPC
- Protocol: HTTPS 443
- Under Health checks > Advanced health checks settings > Success codes, append the code list to read: 200, 303.
- Click Create.

3. Select the target group that you just created, and then click the Targets tab:

- Click Edit
- Select the EC2 instances that are running proxy application, and then click Add to registered.
- Click Save.

4. After the target group is created, you must enable stickiness:

- Open AWS Target Group page (EC2> Load Balancing> Target groups), select the target group instance that you just set up. On the Action menu, select Edit attributes.
- On the Edit attributes page, select Stickiness, specify a duration of 1 day, and then Save changes.

5. On load balancer, update listener rules. Select the load balancer you have configured for this deployment, and then click the Listeners tab:

- For HTTP:80, click View/edit rules. On the resulting Rules page, click the edit icon (once at the top of the page, and then again by the rule) to edit the rule. Delete the existing THEN rule and replace it by clicking Add action > Redirect to.... In the resulting THEN configuration, specify HTTPS and port 443 and leave the other options to default settings. Save the setting and then
Tableau Server Enterprise Deployment Guide

- For HTTP:443, click View/edit rules. On the resulting Rules page, click the edit icon (once at the top of the page, and then again by the rule) to edit the rule.
  
  In the THEN configuration, under Forward to... change the Target group to the HTTPS group that you just created. Under Group-level stickiness, enable stickiness and set duration to 1 day. Save the setting and then click Update.

Step 6: Verify SSL

Verify the configuration by browsing to https://tableau.example.com.

Configure SSL for Postgres

You may optionally configure SSL for the Postgres connection for the external repository connection on Tableau Server.

To simplify certificate management and deployment, and as a security best practice, we recommend using certificates generated by a major trusted-third party certificate authority (CA). Alternatively, you may generate self-signed certificates or use certificates from a PKI for TLS.

This procedure describes how to use OpenSSL to generate self-signed certificate on the Postgres host on a RHEL-like Linux distribution in the example AWS reference architecture.

After you generate and sign the SSL certificate, you must copy the CA certificate to the Tableau host.

On the host running Postgres:

1. Generate signing root certificate authority (CA) key:

   openssl genrsa -out rootCAKey.pem 2048

2. Create the root CA certificate:
openssl req -x509 -sha256 -new -nodes -key rootCAKey.pem -days 3650 -out rootCACert.pem

You will be prompted to enter values for the certificate fields. For example:

Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:Washington
Locality Name (eg, city) [Default City]:Seattle
Organization Name (eg, company) [Default Company Ltd]:Tableau
Organizational Unit Name (eg, section) []:Operations
Common Name (eg, your name or your server's hostname) []:ip-10-0-1-189.us-west-1.compute.internal
Email Address []:example@tableau.com

3. Create the certificate and related key (server-pg.csr and server-pg.key in the example below) for the Postgres computer. The subject name for the certificate must match the EC2 private DNS name of the Postgres host. The subject name is set with the -subj option with the format "/CN=<private DNS name>", for example:

openssl req -new -nodes -text -out server-pg.csr -keyout server-pg.key -subj "/CN=ip-10-0-1-189.us-west-1.compute.internal"

4. Sign the new certificate with the CA certificate that you created in step 2. The following command also outputs the certificate in the crt format:

openssl x509 -req -in server-pg.csr -days 3650 -CA rootCACert.pem -CAkey rootCAKey.pem -CAcreateserial -out server-pg.crt

5. Copy the crt and key files to the Postgres /var/lib/pgsql/12/data/ path:

sudo cp server-pg.crt /var/lib/pgsql/12/data/
sudo cp server-pg.key /var/lib/pgsql/12/data/

6. Switch to root user and set permissions on the cer and key files:

   sudo su

   As root, run the following commands:

   cd /var/lib/pgsql/12/data
   chown postgres.postgres server-pg.crt
   chown postgres.postgres server-pg.key
   chmod 0600 server-pg.crt
   chmod 0600 server-pg.key

   Logout of root:

   exit

7. Update the pg_haba configuration file, /var/lib/pgsql/12/data/pg_hba.conf, to specify md5 trust:

   Change this line: host all all 10.0.1.0/24 trust, to this: host all all 10.0.1.0/24 md5

8. Update the postgresql file, /var/lib/pgsql/12/data/postgresql.conf, by adding this line:

   ssl = on

9. Restart Postgres:

   sudo systemctl restart postgresql-12

Verify SSL connectivity

Before you can verify SSL connectivity, you must copy the certificate that you created in the previous procedure to the client (Tableau host).
On the host where the initial node of Tableau is installed:

1. Copy the CA certificate to the Tableau host:

   ```
   scp ec2-user@<private-DNS-name-of-Postgress-host>:/home/ec2-user/rootCACert.pem /home/ec2-user
   ```

2. Connect to Postgres host over SSL:

   ```
   psql "postgresql://postgres@<IP-address>:5432/ -
   postgres?sslmode=verify-ca&sslrootcert=rootCACert.pem"
   ```

   For example: `psql "postgresql://postgres@10.0.1.189:5432/postgres?sslmode=verify-ca&sslrootcert=rootCACert.pem"

   Postgres will prompt you for the password. After successful sign in, the shell will return:

   ```
   psql (12.5)
   Type "help" for help.
   postgres=#
   ```

Configure SMTP and event notifications

Tableau Server sends email notifications to admins and users. To enable this, you must configure Tableau Server to send mail to your email server. You must also specify the event types, thresholds, and subscription information you want sent.

For the initial configuration of SMTP, and notifications we recommend that you use the configuration file template below to create a json file. You can also set any single configuration key listed below with the syntax described *tsm configuration set* (Linux).

Run this procedure on Node 1 in your Tableau Server deployment:
1. Copy the following json template to a file. Customize the file with your SMTP configuration options and the subscription and alert notifications for your organization.

   - To see a list and description of all SMTP options see SMTP CLI configuration reference (Linux).
   - To see a list and description of all notification event options, see the CLI section of Configure Server Event Notification (Linux).

```json
{
  "configKeys": {
    "svcmonitor.notification.smtp.server": "SMTP server hostname",
    "svcmonitor.notification.smtp.send_account": "SMTP user name",
    "svcmonitor.notification.smtp.port": 443,
    "svcmonitor.notification.smtp.password": "SMTP user account password",
    "svcmonitor.notification.smtp.ssl_enabled": true,
    "svcmonitor.notification.smtp.from_address": "From email address",
    "svcmonitor.notification.smtp.target_addresses": "To email address1,address2",
    "svcmonitor.notification.smtp.canonical_url": "Tableau Server URL",
    "backgrounder.notifications_enabled": true,
    "subscriptions.enabled": true,
    "subscriptions.attachments_enabled": true,
    "subscriptions.max_attachment_size_megabytes": 150,
    "svcmonitor.notification.smtp.enabled": true,
    "features.DesktopReporting": true,
    "storage.monitoring.email_enabled": true,
    "storage.monitoring.warning_percent": 20,
    "storage.monitoring.critical_percent": 15,
  }
}
```
2. Run the `tsm settings import -f file.json` to pass the json file to Tableau Services Manager.

3. Run the `tsm pending-changes apply` command to apply the changes.

4. Run the `tsm email test-smtp-connection` to view and verify the connection configuration.

## Install PostgreSQL driver

To view admin views on Tableau Server, the PostgreSQL driver must be installed on the initial node. The Windows version of Tableau Server Setup installs this driver. However, the PostgreSQL driver is not installed with Tableau Server on Linux.

1. Go to the [Tableau Driver Download](#) page and copy the URL for the PostgreSQL jar file.

2. Run the following procedure on each node of the Tableau deployment:

   - Create the following file path:
     ```bash
     sudo mkdir -p /opt/tableau/tableau_driver/jdbc
     ```

   - From the new path, download the latest version of the PostgreSQL jar file. For example:
     ```bash
     sudo wget https://downloads.tableau.com/drivers/linux/postgresql/postgresql-42.2.22.jar
     ```
3. On the initial node, restart Tableau Server:

```bash
tsm restart
```

## Configure strong password policy

If you are not deploying Tableau Server with an IdP authentication solution, we recommend security hardening the default Tableau password policy.

If you are deploying Tableau Server with an IdP, then you must manage password policies with the IdP.

The following procedure includes JSON configuration for setting password policy on Tableau Server. For more information about the options below, see *Local Authentication* (Linux).

1. Copy the following JSON template to a file. Fill in the key values with your password policy configuration.

```json
{
   "configKeys": {
      "wgserver.localauth.policies.mustcontainletters.enabled": true,
      "wgserver.localauth.policies.mustcontainuppercase.enabled": true,
      "wgserver.localauth.policies.mustcontainnumbers.enabled": true,
      "wgserver.localauth.policies.mustcontainsymbols.enabled": true,
      "wgserver.localauth.policies.minimumpasswordlength.enabled": true,
      "wgserver.localauth.policies.minimumpasswordlength.value": 12,
      "wgserver.localauth.policies.maximumpasswordlength.enabled": false,
   }
}
```
"wgserver.localauth.policies.maximumpasswordlength.value": 255,
"wgserver.localauth.passwordexpiration.enabled": true,
"wgserver.localauth.passwordexpiration.days": 90,
"wgserver.localauth.ratelimiting.maxbackoff.minutes": 60,
"wgserver.localauth.ratelimiting.maxattempts.enabled": false,
"wgserver.localauth.ratelimiting.maxattempts.value": 5,
"features.PasswordReset": true
}
}

2. **Run the `tsm settings import -f file.json` to pass the json file to Tableau Services Manager to configure Tableau Server.**

3. **Run the `tsm pending-changes apply` command to apply the changes.**
Part 7 - Tools and Troubleshooting

Initial node automated recovery

Tableau Server version 2021.2.4 and later include an automated initial node recovery script, `auto-node-recovery`, in the `scripts` directory (`/app/tableau_server-packages/scripts.<version>`).

If there is a problem with the initial node and you have redundant processes on Node 2, there is no guarantee that Tableau Server will continue to run. Tableau Server may continue to run for up to 72 hours after an initial node failure, before the lack of the licensing service impacts other processes. If so, your users may be able to continue to sign in and see and use their content after the initial node fails, but you will not be able to reconfigure Tableau Server because you won’t have access to the Administration Controller.

Even when configured with redundant processes, it is possible that Tableau Server may not continue to function after the initial node fails.

**Note:** If you enabled authorization-to-run (ATR) service during the installation process of Tableau Server, then automated node recovery will not work. You must restore from backup. If you followed the EDG installation process as described in Part 4 of this Guide, then ATR is not enabled in your Tableau deployment.
To recover initial node (Node 1) failure:

1. Sign in to Tableau Server Node 2.

2. Change to the scripts directory:

   ```
   cd /app/tableau_server/packages/scripts.<version>
   ```

3. Run the following command to launch the script:

   ```
   sudo ./auto-node-recovery -p node1 -n node2 -k <license keys>
   ```

   Where `<license keys>` is a comma-separated (no spaces) list of the license keys for your deployment. If you do not have access to your license keys, visit the Tableau Customer Portal to retrieve them. For example:

   ```
   sudo ./auto-node-recovery -p node1 -n node2 -k TSB4-8675-309F-TW50-9RUS,TSNM-559N-ULL6-22VE-SIEN
   ```

   The auto-node-recovery script will execute about 20 steps to recover services to Node 2. Each step is displayed in the terminal as the script progresses. More detailed status is
logged to /data/tableau_data/logs/app-controller-move.log. In most environments, the script takes between 35 and 45 minutes to complete.

Troubleshooting initial node recovery

If node recovery fails, you may find running the script interactively to allow or disallow discrete steps in the process useful. For example, if the script fails part way through the process, you can review log file, make changes to the configuration, and then run the script again. By running in interactive mode, you can then skip all the steps until you get to the step that failed.

To run in interactive mode, add the -i switch to the script argument.

Rebuilding the failed node

After you have run the script, Node 2 will be running all of the services that were formerly on the failed Node 1 host. To add in the 4 node, you need to deploy a fresh Tableau Server host with the bootstrap file and configure it as you did for the original Node 2, as specified in Part 4. See Configure Node 2.

switchto

Switchto is a script from Tim that makes switching between windows easy.

1. Copy the following code into a file called switchto in the home directory on your bastion host.

   #!/bin/bash
   #---------------------------------------------
   -------
   # switchto
   #
   # Helper function to simplify SSH into the various AWS hosts when
# following the Tableau Server Enterprise Deployment Guide (EDG).
#
# Place this file on your bastion host and provide your AWS hosts' internal ip addresses or machine names here.
# Example: readonly NODE1="10.0.3.187"
#
readonly NODE1=""
readonly NODE2=""
readonly NODE3=""
readonly NODE4=""
readonly PGSQL=""
readonly PROXY1=""
readonly PROXY2=""

usage() {
    echo "Usage: switchto.sh [ node1 | node2 | node3 | node4 | pgsql | proxy1 | proxy2 ]"
}

ip=""

case $1 in
    node1)
        ip="$NODE1"
        ;;
    node2)
        ip="$NODE2"
        ;;
    node3)
        ip="$NODE3"
Tableau Server Enterprise Deployment Guide

```
;;
node4)
ip="$NODE4"
;;
pgsql)
ip="$PGSQL"
;;
proxy1)
ip="$PROXY1"
;;
proxy2)
ip="$PROXY2"
;;
?)
usage
exit 0
;;
*)
  echo "Unknown option $1."
  usage
  exit 1
  ;;

esac

if [[ -z $ip ]]; then
  echo "You must first edit this file to provide the IP addresses of your AWS hosts."
  exit 1
fi

ssh -A ec2-user@$ip
```

2. Update the IP addresses in the script to map to your EC2 instances and then save the file.
3. Apply permissions to the script file:
Usage:

To switch to a host, run the following command:

`.switchto <target>`

For example, to switch to Node 1, run the following command:

`.switchto node1`
Appendix - AWS Deployment Toolbox

This topic includes tools and alternate deployment options for the reference architecture when deployed in AWS. Specifically, this topic describes how to automate the example AWS deployment that is described throughout the EDG.

EDGSetup automated installation script

The EDGSetup script automates the implementation of the four-node Tableau deployment that is described in Part 4 - Installing and Configuring Tableau Server. If you are following the example AWS implementation as described in this Guide, then you may be able to run EDGSetup. EDGSetup does not support the three Availability Zone five-node deployment. However, you can run EDGSetup, and then install Node 5 manually as specified in Part 4.

Requirements. To run the script, you must prepare and configure the AWS environment according to the example implementation in Part 3 - Preparing for Tableau Server Enterprise Deployment:

- VPC, subnet, and security groups have been configured as described. IP addresses do not have to match those that are shown in the example implementation.
- Four EC2 instances running the latest, updated builds of AWS Linux 2
- PostgreSQL is installed and has been configured as described in Install, configure, and tar PostgreSQL.
- A Step 1 tar backup file is on the EC2 instance where PostgreSQL is installed, as described in Take PostgreSQL Step 1 tar backup.
- The EC2 instance that will be running Node 1 of the Tableau Server deployment has been configured to communicate with PostgreSQL as described in Part 4 - Installing and Configuring Tableau Server.
- You have logged into each EC2 instance with an SSH session from the bastion host.
The script takes about 1.5-2 hours to install and configure the four Tableau servers. The script configures Tableau according to the prescribed settings of the reference architecture.

The script performs the following actions:

- Restores Stage 1 backup of PostgreSQL host if you specify a path to the PostgreSQL host's tar file.
- Obliterates existing Tableau installations on all nodes.
- Runs `sudo yum update` on all nodes.
- Downloads and copies Tableau rpm package to each node.
- Downloads and installs dependencies to each node.
- Creates `/app/tableau_server` and installs package on all nodes.
- Installs Node 1 with a local identity store and configures external repository with PostgreSQL.
- Performs bootstrap installation and initial configuration of Node 2- Node 4.
- Deletes the bootstrap file and the configuration file for EDGSetup.
- Configures services across the Tableau cluster according to reference architecture specifications.
- Validates installation and returns status for each node.

**Download and copy the script to the bastion host**

1. Copy the script from the [EDGSetup.txt samples page](#) and paste the code into a file called, `EDGSetup`.
2. Save the file to the home directory on the EC2 host that is serving as the bastion host.
3. Run the following command to change the mode on the file to make it executable:

   ```bash
   sudo chmod +x EDGSetup
   ```

**Run EDGSetup**

EDGSetup must be run from the bastion host. The script has been written with the assumption that you are running under the context of ssh forward agent as described at Example: Connect to bastion host in AWS. If you are not running with ssh forward agent context, then you will be prompted for passwords throughout the installation process.
1. Create, edit, and save a registration file (registration.json). The file must be a properly-formatted json file. Copy and customize the following template:

```json
{
    "zip" : "97403",
    "country" : "USA",
    "city" : "Springfield",
    "last_name" : "Simpson",
    "industry" : "Energy",
    "eula" : "yes",
    "title" : "Safety Inspection Engineer",
    "phone" : "5558675309",
    "company" : "Example",
    "state" : "OR",
    "department" : "Engineering",
    "first_name" : "Homer",
    "email" : "homer@example.com"
}
```

2. Run the following command to generate a template configuration file:

   ```bash
   ./EDGSetup -g edg.config
   ```

3. Open the configuration file to edit:

   ```bash
   sudo nano edg.config
   ```

   At a minimum, you must add the IP addresses of each EC2 host, a file path to the registration file, and a valid license key.

4. When you are done editing the configuration file, save it, and then close it.

5. To run EDGSetup, run the following command:

   ```bash
   ./EDGSetup -f edg.config
   ```