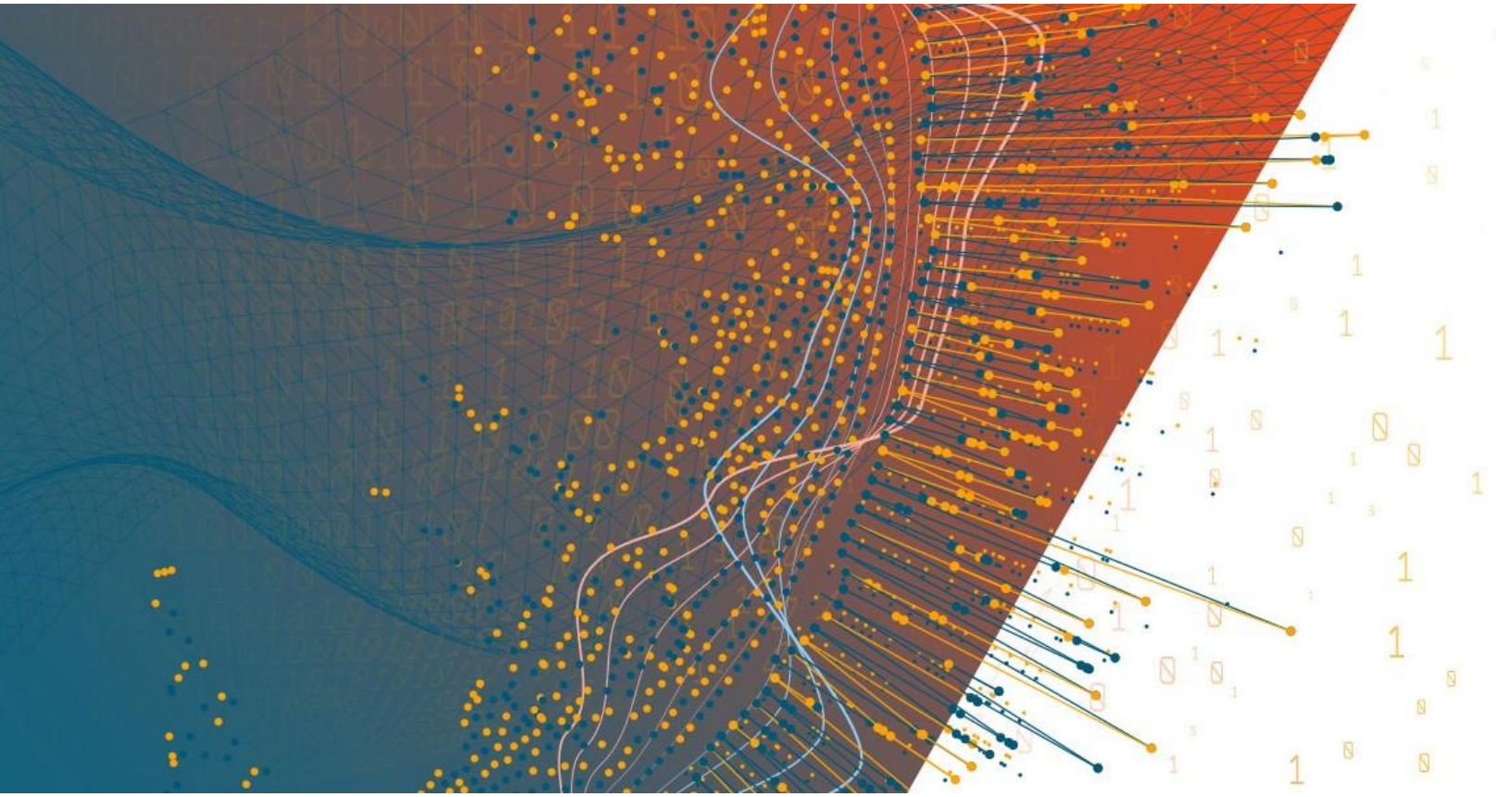

ALTAIR

Altair Panopticon™ v2025.1

K3S DEPLOYMENT OF PANOPTICON

TABLE OF CONTENTS

INTRODUCTION

Some key benefits of Kubernetes deployment of Panopticon include:

- Multi-tenancy support
- Self-healing
- Horizontal and Vertical scaling
- Standardized and secure application setup

K3s, a lightweight Kubernetes implementation compared to K8s, is suitable for cloud-neutral and on-premises deployment, as opposed to managed Kubernetes solutions offered by all major cloud providers (i.e., AWS, Azure, GCP, OCI).

Both parts in deploying Panopticon on K3s are discussed in this guide:

- K3s installation on the host
- Panopticon installation

INSTALLATION

Altair provides an installation package with the following content:

For K3s Installation

Resource	Description
node_setup.sh	This script is to be run on every node that will be added to the K3s cluster.
master_setup.sh	This shell script is to be run exclusively on master node in the K3s cluster. **NOTE:** Ensure to run `node_setup.sh` script first.
worker_setup.sh	This shell script is to be run on every worker node added to the K3s cluster. **NOTE:** Ensure to run `node_setup.sh` script first.

For Panopticon Installation

Resource	Description
pano_repo_setup.sh	This script adds the Altair repository to helm and pulls the helm chart as per the `PANO_HELM_VER` entry in the `pano.env` file. This shell script is to be run exclusively on master node in the K3s cluster.
cert_gen.sh	This shell script is to be run exclusively on master node in the K3s cluster. **NOTE:** This script is to be run only for generating a self-signed certificate for demo/PoC. For production setup, the .CRT and key file/values and the domain should be provided by the IT department of your organization.
pano.env	This file contains various properties as key-value pairs to control the application/helm version to be installed, the Altair repository URLs, etc.

SINGLE NODE DEPLOYMENT

The instructions below assume that an Ubuntu Server host is used.

INSTALLING THE K3S CLUSTER

Steps:

1. Through Secure Shell (SSH), connect to the host.
2. Extract the contents of the installation package and CD to the directory source (i.e., Ubuntu).
3. Switch to super user using the following command:

   ```
   sudo su
   ```

4. Ensure that script files are executable using the following command:

   ```
   chmod 555 node_setup.sh
   ```

5. Do the same for `master_setup` and other scripts.
6. Run the `node_setup.sh` script:

   ```
   ./node_setup.sh
   ```

7. Run the `master_setup.sh` script:

   ```
   ./master_setup.sh
   ```

8. Once completed, verify that all pods are running/completed using the following command:

   ```
   kubectl get pods -A
   ```

   Here is a sample output:

```
NAMESPACE          NAME                                                READY   STATUS
kube-system        local-path-provisioner-6c86858495-2nh82             1/1     Running
kube-system        coredns-6799fbcd5-6vgvp                             1/1     Running
kube-system        helm-install-traefik-crd-lcf4m                      0/1     Completed
kube-system        svclb-traefik-4be08f3a-p6v8g                        2/2     Running
kube-system        helm-install-traefik-pstvc                          0/1     Completed
kube-system        metrics-server-54fd9b65b-zzjll                      1/1     Running
kube-system        traefik-f4564c4f4-7dshg                             1/1     Running
longhorn-system    longhorn-ui-7d4b94df76-6fprs                        1/1     Running
longhorn-system    longhorn-ui-7d4b94df76-7qcrd                        1/1     Running
longhorn-system    longhorn-manager-5wrnz                              1/1     Running
longhorn-system    longhorn-driver-deployer-576d574c8-srz7g            1/1     Running
longhorn-system    engine-image-ei-acb7590c-9k4kg                      1/1     Running
longhorn-system    instance-manager-5165d609d0cf2cbb7fa19a7e084f1814   1/1     Running
longhorn-system    csi-provisioner-667796df57-htjsk                    1/1     Running
longhorn-system    csi-provisioner-667796df57-bbdxp                    1/1     Running
longhorn-system    csi-provisioner-667796df57-5jlbm                    1/1     Running
longhorn-system    csi-snapshotter-959b69d4b-k57xp                     1/1     Running
longhorn-system    csi-attacher-5c4bfdcf59-4b7xs                       1/1     Running
longhorn-system    csi-snapshotter-959b69d4b-6wljg                     1/1     Running
longhorn-system    csi-attacher-5c4bfdcf59-flth6                       1/1     Running
longhorn-system    csi-attacher-5c4bfdcf59-4tvxt                       1/1     Running
longhorn-system    csi-snapshotter-959b69d4b-bdw87                     1/1     Running
longhorn-system    csi-resizer-694f8f5f64-h72rl                        1/1     Running
longhorn-system    csi-resizer-694f8f5f64-n99tr                        1/1     Running
longhorn-system    csi-resizer-694f8f5f64-q2v5c                        1/1     Running
               longhorn-system   longhorn-csi-plugin-zpz44                          3/3
               Running
```

INSTALLING PANOPTICON ON THE K3S CLUSTER

Steps:

1. Through SSH, connect to the host where K3s has been installed.
2. To generate the self-signed certificate that can be used for demo, testing, and Proof of Concept (POC) projects, run `cert_gen.sh` for the fake domain **pano.k3s.test.com** using the following command:

   ```
   ./cert_gen.sh
   ```

   This produces the `pano_certs` folder with `tls_crt.out` and `tls_key.out` files.
3. Run the `pano_repo_setup.sh` script using the following command:

   ```
   ./pano_repo_setup.sh
   ```

   This produces the `pano_charts` folder with a .ZIP file. For example:

   ```
   panopticon-0.2.38-master.24.0.0.33210.12.c29d835.f76596c.tgz
   ```

   The downloaded helm chart depends on the value of `PANO_HELM_VER` in the `pano.env` file.
4. Extract the contents of the .TGZ file into a folder named **panopticon**. Move into the folder using the following command:

   ```
   cd panopticon
   ```

   The folder contents when running the `ls` command should look like:

   ```
   Chart.yaml Jenkinsfile README.md Readme.txt Version.properties templates
   values.yaml
   ```
5. Edit `values.yaml` with the details below and then save the file:

   ```
   cloud_type: k3s
   authentication_mode: local
   license:
     hwu_uri: <license server URI>
   cert_settings:
     use_external_cert: true
     external_url: pano.k3s.test.com
     tls_crt: <Contents of tls_crt.out from running the cert_gen.sh to be
   copied here>
     tls_key: <Contents of tls_key.out from running the cert_gen.sh to be
   copied here>
   ```

NOTE	Please see comment lines inside the `values.yaml` file for further guidance.

6. Run the following command to produce a `final.yaml` deployment descriptor:

   ```
   helm template . > final.yaml
   ```
7. Run the following command to deploy this deployment descriptor to K3s:

   ```
   kubectl apply -f final.yaml
   ```
8. Confirm all pods are running using the command:

   ```
   kubectl get pods -n pano-test1
   ```

ACCESSING PANOPTICON FROM A CLIENT MACHINE

Since the fake (test) domain `pano.k3s.test.com` will not be resolved by any DNS, you need to add the entry to the hosts file in the client machine. The entry in the host file should look like this example:

```
101.102.103.104 pano.k3s.test.com
```

The IP address should be the public IP address of the machine where you are running the K3s master node.

On Linux, the hosts file can be found here: `/etc/hosts`

On Windows, the hosts file can be found here: `C:/Windows/System32/drivers/etc/hosts`

Afterward, you should be able to reach the Panopticon server running on K3s on this URL: https://pano.k3s.test.com.

FETCHING DEFAULT USER CREDENTIALS FOR LOCAL USERS FROM USERS.XML

Using the local `authentication_mode` creates a set of users and passwords to be able to log on to Panopticon.

The login information from the local configuration can be obtained by running the following command:

```
kubectl get configmaps/pano-users-conf -n pano-test1 -o yaml
```

ADDING JDBC DRIVERS

Follow the instructions below to add the JDBC drivers.

Steps:

1. Run the following command into the pod container:

   ```
   sudo kubectl exec -it statefulset-vizapp-0 /bin/bash -n pano-test1
   ```

2. Run the following command to go to the `appdata` folder.

   ```
   cd /etc/panopticon/appdata
   ```

3. Create the `jar_extentions` folder by running this command:

   ```
   mkdir jar_extentions
   ```

NOTE	The folder must be named `jar_extentions`.

   For example:

```
pano@statefulset-vizapp-0:/etc/panopticon/appdata$ mkdir jar_extentions
pano@statefulset-vizapp-0:/etc/panopticon/appdata$ ls -all
total 104
drwxrwsr-x 13 root pano  4096 Feb 24 16:13 .
drwxr-xr-x  1 root root  4096 Dec  5 14:08 ..
drwxrws---  2 pano pano  4096 Feb 21 21:00 .cache
drwxrws---  3 pano pano  4096 Feb 19 20:35 CacheData
drwxrws---  3 pano pano  4096 Feb 19 20:35 default-settings
drwxr-sr-x  2 pano pano  4096 Feb 24 16:13 jar_extentions
drwxrws---  2 pano pano  4096 Feb 19 20:36 JavaScriptConfiguration
-rw-rw----  1 pano pano    13 Feb 19 17:50 last_version
drwxrws---  2 root pano 16384 Feb 19 17:49 lost+found
-rw-rw-r--  1 pano pano  5759 Feb 24 14:16 Panopticon.properties
-rw-rw-r--  1 pano pano  5759 Feb 24 14:16 Panopticon.properties.bak
-rw-r-----  1 pano pano    17 Feb 24 14:17 Parameters.json
drwxrws---  4 pano pano  4096 Feb 20 15:28 .repository
-rw-r--r--  1 root pano   154 Feb 24 14:16 security.yml
drwxrws---  4 pano pano  4096 Feb 20 15:30 shared
drwxrws---  2 pano pano  4096 Feb 19 20:36 Sounds
drwxrws---  2 pano pano  4096 Feb 24 16:13 Statistic
-rw-rw----  1 pano pano    64 Feb 19 20:35 token_secret
drwxrws---  3 pano pano  4096 Feb 20 15:28 UserData
-rw-r--r--  1 root pano  1109 Feb 24 14:16 users.xml
```

4. Run the following command to exit from the shell within the container and come back to the host machine from which you are expected to run step 5:

   `exit`

5. Copy your JAR file onto the machine running K3s.

6. Copy the JAR file from the machine running K3s into the `jar_extentions` folder using the following command:

   `sudo kubectl cp <source> <pod-name>:<path> -n <namespace>`

   For example, you have the following entries for the command line:

   - Pod is `statefulset-vizapp-0`
   - Namespace is `pano-test1`
   - The JAR file to be copied is `mssql-jdbc-12.6.1.jre8.jar`
   - Location of the JAR file to be copied is `/home/tony/pano-k3s-install`
   - Destination folder is `/etc/panopticon/appdata/jar_extentions`

   The command line would be:

   ```
   sudo kubectl cp /home/tony/pano-k3s-install/mssql-jdbc-12.6.1.jre8.jar
   statefulset-vizapp-0:/etc/panopticon/appdata/jar_extentions/mssql-jdbc-
   12.6.1.jre8.jar -n pano-test1
   ```

7. Restart the pod using the following command:

   `sudo kubectl delete pod/statefulset-vizapp-0 -n pano-test1`

MULTI-NODE DEPLOYMENT

It is possible to deploy K3s with one or more worker nodes to allow scaling. Follow the steps below to deploy these scenarios.

MASTER NODE SETUP

Steps:

1. Through SSH, connect to the master node.
2. Switch to super user using the following command:

   `sudo su`
3. Open the `pano.env` file and set the following property:

   `CLUSTER_TYPE=MULTI_NODE`
4. Run `node_setup.sh`.
5. Run `master_setup.sh` and note down the master IP and the cluster token

WORKER NODE SETUP

Steps:

1. Through SSH, connect to the worker node(s).
2. Ensure to copy the `worker_setup.sh` script.
3. Switch to super user using the following command:

   `sudo su`
4. Run `node_setup.sh`.
5. Run `worker_setup.sh <master_ip> <k3s_token>`.

   Where:

   - `master_ip` : Can be taken from `hostname -i` command on the master node
   - `k3s_token`: Can be taken from `/var/lib/rancher/k3s/server/node-token` on the master node
6. Check that the worker node is added to the cluster with command `kubectl get nodes` to produce output like below:

```
root@panopticon-dev-k3s-master:/home/shashil# kubectl get nodes
NAME                          STATUS   ROLES                  AGE
VERSION
panopticon-dev-k3s-master     Ready    control-plane,master   29h
v1.28.8+k3s1
panopticon-dev-k3s-worker     Ready    <none>                 4h53m
v1.28.8+k3s1
```

The Panopticon application deployment steps remain the same with single node clusters.

UNINSTALLING

In case you need to uninstall and reinstall a specific deployment of Panopticon or the entire K3s cluster, you can follow these steps:

- To delete a specific Panopticon deployment, run the following command:

  ```
  kubectl delete -f final.yaml
  ```

NOTE	This will delete all components that got installed via the `kubectl apply` command.

- To delete the entire K3s cluster itself from the master node, you can run the following command:

  ```
  /usr/local/bin/k3s-uninstall.sh
  ```

NOTE	This will delete the Panopticon deployment and the K3s cluster from the underlying machine.

- To delete the entire K3s cluster itself from worker node, you can run the following command:

  ```
  /usr/local/bin/k3s-agent-uninstall.sh
  ```

03.2025

ABOUT PANOPTICON

For more information on Panopticon and other resources, go to https://www.altair.com/panopticon.