K3S

Contains information related to the K3S and the application system

- Get Statefulsets
- get user names and passwords for applications
- K3S applications certificates guide
- K3S Commands
- Start and Stop using HeavyScript

Get Statefulsets

sudo k3s kubectl get statefulsets -A | grep "ix-"

get user names and passwords for applications

Use this script in a .sh format to get application passwords

#!/bin/bash

```
# get namespaces
namespaces=$(k3s kubectl get secrets -A | grep -E "dbcreds|cnpg-main-urls" | awk '{print $1, $2}')

# iterate over namespaces
( printf "Application | Username | Password | Address | Port\n"
echo "$namespaces" | while read ns secret; do
    # extract application name
    app_name=$(echo "$ns" | sed 's/^ix-//')
    if [ "$secret" = "dbcreds" ]; then
        creds=$(k3s kubectl get secret/$secret --namespace "$ns" -o jsonpath='{.data.url}' | base64
-d)
    else
        creds=$(k3s kubectl get secret/$secret --namespace "$ns" -o jsonpath='{.data.std}' | base64
-d)
    fi

    # get username, password, addresspart, and port
    username=$(echo "$creds" | awk -F '//' '{print $2}' | awk -F ':' '{print $1}')
    password=$(echo "$creds" | awk -F ':' '{print $3}' | awk -F '@' '{print $1}')
```

```
    addresspart=$(echo "$creds" | awk -F '@' '{print $2}' | awk -F ':' '{print $1}')
    port=$(echo "$creds" | awk -F ':' '{print $4}' | awk -F '/' '{print $1}')

    # construct full address
    full_address="${addresspart}.${ns}.svc.cluster.local"

    # print results with aligned columns
    printf "%s | %s | %s | %s | %s\n" "$app_name" "$username" "$password" "$full_address" "$port"
done ) | column -t -s "|"
```

K3S applications certificates guide

`sudo k3s kubectl get challenges -n ix-appname`

`sudo k3s kubectl describe challenge CHALLENGENAME -n ix-appname`

K3S Commands

`k3s crictl pull <imageName>`

`k3s kubectl get pods -n <ix-appname>`

`k3s kubectl get pods --all-namespaces`

`k3s kubectl get <appname>`

`sudo k3s kubectl get events -n ix-APPNAME`

1. **Check Cluster Health:**
   `k3s kubectl get nodes`
   This command checks the health and status of all nodes in the cluster.
2. **List All Pods:**
   `k3s kubectl get pods --all-namespaces`
   This command lists all pods across all namespaces, which can help identify any pods that are not running as expected.
3. **Describe a Pod:**
   `k3s kubectl describe pod <pod-name> -n <namespace>`
   Replace `<pod-name>` with the name of the pod and `<namespace>` with the namespace it resides in. This command provides detailed information about a specific pod, which can be useful for diagnosing issues.
4. **View Logs for a Pod:**
   `k3s kubectl logs <pod-name> -n <namespace>`
   This command shows the logs for a specific pod. If the pod has multiple containers, you may need to specify the container name with the `-c` flag.
5. **Check Deployments:**
   `k3s kubectl get deployments -n <namespace>`
   This command lists all deployments in a specific namespace, showing their desired and current states.
6. **Describe a Deployment:**
   `k3s kubectl describe deployment <deployment-name> -n <namespace>`
   This command provides detailed information about a specific deployment.
7. **Check Services:**

`k3s kubectl get svc -n <namespace>`
This command lists all services in a specific namespace, which can help troubleshoot networking and access issues.

8. **Check Persistent Volume Claims (PVCs):**
   `k3s kubectl get pvc -n <namespace>`
   This command lists all PVCs in a specific namespace, which can help troubleshoot storage-related issues.

9. **Check ConfigMaps:**
   `k3s kubectl get configmaps -n <namespace>`

This command lists all ConfigMaps in a specific namespace, which can be useful for troubleshooting configuration issues.

10. **Check Ingress Resources:**
    `k3s kubectl get ingress -n <namespace>`
    This command lists all ingress resources in a specific namespace, which can help troubleshoot external access to services.

11. **Delete a Pod:**
    `k3s kubectl delete pod <pod-name> -n <namespace>`
    This command deletes a specific pod. Kubernetes will attempt to recreate the pod if it is managed by a controller like a Deployment or StatefulSet.

12. **Scale a Deployment:**
    `k3s kubectl scale deployment <deployment-name> -n <namespace> --replicas=<number>`
    Replace `<number>` with the desired number of replicas. This command adjusts the number of replicas for a deployment, which can be useful for scaling up or down.

Remember to replace placeholders like `<pod-name>`, `<namespace>`, `<deployment-name>`, and `<number>` with actual values relevant to your environment. Also, please note that manually scaling or deleting Kubernetes objects can have unintended consequences, especially if they are managed by Helm or other automation tools. Always ensure you understand the impact of these commands before executing them.

Start and Stop using HeavyScript

Start: heavyscript app -s <applicationName>
Stop: heavyscript app -x <applicationName>