OpenShift Plattform Monitoring Skript

This script will run some basic tests on a given openshift platform. The eintension is to have a first state of the platform after having done updates, etc.

How to Use

Bei jedem Start wird eine kurze Beschreibung Anleitung angezeigt. Man muss sich jedes Mal auf der entsprechenden OSE Plattform einloggen. Es braucht für den Betrieb ein gültiges Host Inventory File. Es werden Angaben zum Login etc aus dem angegebenen Host Inventory File ausgelesen und angewendet, - so z.B. openshift_logging_master_public_url.

```
u229044@oseadmin01az1 admin$ ./ose_monitoring_checks.sh
../ansible/inventory/tss/
============================================================
This script does some basic checks of a given OSE platform.

you need to enter the path to the host inventory file

Please be aware, that this script is only to be run on the two
bastion hosts oseadmin01 and 02 from your personal account.
You need OSE cluster admin privileges to run this script
Please keep in mind, that your local ~/.kube/config will
be erased on each run. The reason is to make sure you are
working only on the OSE cluster of the given host file
This is also the reason why you have to oc login on each run

If you entcounter many ssh missing fingerprint messages:
EXECUTE: # export ANSIBLE_HOST_KEY_CHECKING=False

============================================================
Please enter ansible vault password:
Getting master cluster public hostname from ansible..
Log in on: https://master.ose.sbb-cloud.net:8443
Authentication required for https://master.ose.sbb-cloud.net:8443
(openshift)
Username: USER
Password: PASSWORD

Login successful.
. . .
```

Skript Sourcecode

Filename: ose_monitoring_checks.sh

```
#!/bin/bash
#
# abstract: This script will run some basic tests on a given openshift
#           platform. Th eintension is to have a first state of the
#           platform after having done updates, etc.
#
#           To be able to have a defined context, when working on the
bastion host,
#           one needs the input of a hosts inventory file.
#
#
# run this script on bastion host
#
# author:   M.Reber - Cloud Platform Team
# date:     2019.08.19
#=================================================================
#
#=================================================================
#
tmpfile=$(mktemp)
ansible_vault=$(mktemp)
#
# if script is interupted execute the cleanup function
trap 'cleanup_function' exit
# Function that gets executed on script interruption.
cleanup_function () {
        # cleanup
        rm -f ${tmpfile}
        rm -f ${ansible_vault}
}
#
echo "============================================================"
echo "This script does some basic checks of a given OSE platform."
echo ""
echo "you need to enter the path to the host inventory file"
echo ""
echo "Please be aware, that this script is only to be run on the two"
echo "bastion hosts oseadmin01 and 02 from your personal account."
echo "You need OSE cluster admin privileges to run this script"
echo "Please keep in mind, that your local ~/.kube/config will"
echo "be erased on each run. The reason is to make sure you are"
echo "working only on the OSE cluster of the given host file"
echo "This is also the reason why you have to oc login on each run"
echo ""
```

```
echo "If you entcounter many ssh missing fingerprint messages:"
echo "EXECUTE: # export ANSIBLE_HOST_KEY_CHECKING=False"
echo ""
echo "============================================================="
#
#
#-------------------------------------------------------------------------
---------
#                            INITIAL SKRIPT ENV CHECKS:
#-------------------------------------------------------------------------
---------
# check if the input parameter is the host inventory file of the platform
invfile="${1}"
if [[ ! -d "${invfile}" ]] ; then
  echo "Usage $0 ../ansible/inventory/otc_test03/"
  exit 1
fi
if ! hostname --short | grep --silent oseadmin ; then
  echo "Run this script on bastion host oseadmin01 or oseadmin02 only, exit"
  exit 1
fi
#
echo Please enter ansible vault password:
read -s ansible_vault_pw
echo $ansible_vault_pw > $ansible_vault
#
echo "Getting master cluster public hostname from ansible.."
master_url=https://$(ansible masters -i ${invfile} -m debug -a
"var=openshift_master_cluster_public_hostname" --vault-password-
file=${ansible_vault} | grep -v SUCCESS | awk '{print $2}' | head -1 | sed
's/"//g'):8443
#
# we need to get the cluster name out of the master_url and there
# replace the "." (dot) with "-" (dash)
# there is also need to cut the https:// off
cluster_name=$(echo $master_url | tr . - | cut -d"/" -f3)
#
# this script should only run as unprivileged user
if [[ "$EUID" -eq 0 ]]; then
  echo "Error, run this script as unprivileged user."
  exit 1
fi
#
# here we log in to the cluster
echo "Log in on: ${master_url}"
oc login ${master_url} || exit $?
#
clear
#-------------------------------------------------------------------------
---------
#                            START WITH CLUSTER CHECKS:
```

```
#--------------------------------------------------------------------------
---------
#
# check openshift_logging_master_public_url
echo "==== Checking master_public_url health=="
health=$(curl --silent ${master_url}/healthz)
if [[ "${health}" != "ok" ]] ; then
  echo "[ NOK ] master_url ${master_url} health problem ${health}"
else
  echo "[ OK ] master_url ${master_url}  health is ${health}"
fi
echo ""
#
echo "==== Checking cluster globaly ==="
cmd_status=`ansible masters -i ${invfile} -m shell -a "ps -ef | grep etcd |
grep -v grep" --vault-password-file=${ansible_vault} | grep -v SUCCESS |
grep -v etcd`
if [[ "${cmd_status}" != "" ]] ; then
  echo "[ NOK ] not all etcd on masters present: ${cmd_status}"
else
  echo "[ OK ] all etcd processes present"
fi
#
first_master=$(ansible masters -i ${invfile} --list-host --vault-password-
file=${ansible_vault} | grep -v hosts | head -n1 | awk '{print $1}')
master_nodes=$(ansible ${first_master} -i ${invfile} -m shell -a "oc get
nodes | grep master" --vault-password-file=${ansible_vault} | grep -v
SUCCESS | awk '{print $1}')
#
# here we check the health status of the etcd cluster
# this can be done by selecting the etcd container and passing
etccmd_string=""
for node in ${master_nodes}
  do
    if [[ "${etccmd_string}" == "" ]] ; then
      etccmd_string="https://${node}:2379"
    else
      etccmd_string="${etccmd_string},https://${node}:2379"
    fi
done
etccmd_string="${etccmd_string} --ca-file=/etc/etcd/ca.crt --key-
file=/etc/etcd/peer.key --cert-file=/etc/etcd/peer.crt cluster-health"
#
# Checks OpenShift Cluster minor version.. 3.X. (Workaround if OpenShift is
on 3.9..)
if [[ $(ansible ${first_master} -i ${invfile} -m shell -a "oc version | grep
openshift" --vault-password-file=${ansible_vault} | grep -v SUCCESS | awk
'{print $2}' | cut -d '.' -f2) != "9" ]] ; then
  # get docker container ID
  docker_id=$(ansible ${first_master} -i ${invfile} -m shell -a "docker ps |
```

```
grep etcd_master-etcd" --vault-password-file=${ansible_vault} | grep -v
SUCCESS| awk '{print $1}')
  clus_stat=$(ansible ${first_master} -i ${invfile} -m shell -a "docker exec
-t ${docker_id} etcdctl -C ${etccmd_string}" --vault-password-
file=${ansible_vault} | grep cluster | grep healthy)
#
  if [[ "${clus_stat}" == "" ]] ; then
    # we got a negative result, cluster is not healthy
    echo "[ NOK ] etcd cluster does not show healthy result on master
${first_master}"
    ansible ${first_master} -i ${invfile} -m shell -a "docker exec -t
${docker_id} etcdctl -C ${etccmd_string}" --vault-password-
file=${ansible_vault}
  else
    echo "[ OK ] etcd cluster is healthy on master ${first_master}"
  fi
else
  echo "In OpenShift 3.9 etcd service is a systemd service (not
dockerised..) - checking service status"
  clus_stat=$(ansible ${first_master} -i ${invfile} -m shell -a "systemctl
status etcd" --vault-password-file=${ansible_vault} | grep -v SUCCESS | grep
"Active:" | awk '{print $2}')
  if [[ "${clus_stat}" == "active" ]] ; then
    echo "[ OK ] etcd cluster is healthy on master ${first_master}"
  else
    echo "[ NOK ] etcd cluster does not show healthy result on master
${first_master} error was:"
    ansible ${first_master} -i ${invfile} -m shell -a "systemctl status
etcd" --vault-password-file=${ansible_vault} | grep -v SUCCESS
  fi
fi
#
# here we check the reachability of all nodes
cmd_status=$(ansible nodes -i ${invfile} --timeout=40 -a 'echo "success"' --
vault-password-file=${ansible_vault} | grep -vi SUCCESS)
if [[ "${cmd_status}" != "" ]] ; then
  echo "[ NOK ] not all nodes reachable: ${cmd_status}"
else
  echo "[ OK ] all nodes reachable by ansible and ssh"
fi
cmd_status=$(ansible ${first_master} -i ${invfile} -m shell -a "oc get nodes
| grep -v Ready" --vault-password-file=${ansible_vault} | grep -v SUCCESS |
grep -v NAME)
if [[ "${cmd_status}" != "" ]] ; then
  echo "[ NOK ] review nodes, not all nodes are shedulable: ${cmd_status}"
  ansible ${first_master} -i ${invfile} -m shell -a "oc get nodes | grep -v
Ready" --vault-password-file=${ansible_vault} | grep -v SUCCESS
else
  echo "[ OK ] all nodes present and ready"
fi
cmd_status=$(oc get pods --all-namespaces | grep -v Running |grep -v
```

```
Completed | grep -v NAMESPACE)
if [[ "${cmd_status}" != "" ]] ; then
  echo "[ NOK ] there are Pods in not-ok status, review"
  oc get pods --all-namespaces | grep -v Running | grep -v Completed
  pod_errors=true
  fixable_errors=true
else
  echo "[ OK ] all Pods running OK"
fi
echo ""
#
echo "==== Checking ha proxy pod presence  ==="
cmd_status=$(oc get pods --no-headers -n default | grep -v "1/1")
if [[ "${cmd_status}" != "" ]] ; then
  echo "[ NOK ] review ha pods not all running: ${cmd_status}"
  oc get pods --no-headers -n default | grep -v "1/1"
else
  echo "[ OK ] all needed Pods are runnig as desired READY 1/1"
fi
echo ""
#
echo "==== Checking HA Router Status ==="
# we need to have all DC running the same number desired and current
#oc get dc -n default
for res in $(oc get dc --no-headers -n default | awk '{print $1}');do
  desnr=$(oc get dc --no-headers -n default | grep "^${res}" | awk '{print
$3}')
  currnr=$(oc get dc --no-headers -n default | grep "^${res}" | awk '{print
$4}')
  if [[ ${desnr} -ne ${currnr} ]] ; then
    echo "[ NOK ] not equal DESIRED ${desnr} and CURRENT ${currnr} of DC Name
${res}"
  else
    echo "[ OK ] runnig DESIRED ${desnr} and CURRENT ${currnr} of DC Name
${res}"
  fi
done
#
# here we need to check the image version if it is on the same level as the
RPM package
package_version=$(ansible ${first_master} -i ${invfile} -m shell -a
"warn=false rpm -qa | grep atomic-openshift-node | cut -d '-' -f4" --vault-
password-file=${ansible_vault} | grep -v 'SUCCESS')
echo "Checking image version of ha-router"
router_name=$(oc get dc -n default | grep router | head -1 | awk '{print
$1}')
oc get dc -n default ${router_name} -o yaml | grep "image:"
check_vers=$(oc get dc -n default ${router_name} -o yaml | grep "image:" |
grep ${package_version})
if [[ "${check_vers}" == "" ]] ; then
```

```
  echo "[ NOK ] OSE version of HA router is not equal like version
${package_version} in hosts file"
else
  echo "[ OK ] OSE version of HA router is equal like version
${package_version} in hosts file"
fi
echo ""
#
echo "==== Checking openshift-monitoring ==="
# OSE monitoring need to be running on each node
# on each node we have a node-exporter
# we have one pod alertmanager-main-1 and two prometheus-k8s
oc get pods -n openshift-monitoring -o wide | sort -k 7 > ${tmpfile}
for node in $(ansible ${first_master} -i ${invfile} -m shell -a "oc get
nodes --no-headers" --vault-password-file=${ansible_vault} | grep -v SUCCESS
| awk '{print $1}')
  do
    podfnd=$(grep ${node} ${tmpfile} | grep node-exporter| grep "2/2")
    if [[ "${podfnd}" == "" ]] ; then
      echo "[ NOK ] pod node-exporter is not running 2/2 on node ${node}"
    else
      echo "[ OK ] pod node-exporter is running 2/2 on node ${node}"
    fi
done
echo ""
prom_master_ste_1=$(grep prometheus-k ${tmpfile}| awk '{print $2}'| cut -
d"/" -f1)
prom_master_ste_2=$(grep prometheus-k ${tmpfile}| awk '{print $2}'| cut -
d"/" -f1)
prom_master_node=$(grep prometheus-k ${tmpfile}| awk '{print $7}')
if [[ "${prom_master_ste_1}" != "${prom_master_ste_2}" ]] ; then
  echo "[ NOK ] pod prometheus-k8s is not running as desired
${prom_master_ste_1}/${prom_master_ste_2} on node ${prom_master_node}"
else
  echo "[ OK ] pod prometheus-k8s is running as desired
${prom_master_ste_1}/${prom_master_ste_2} on node ${prom_master_node}"
fi
rm -f ${tmpfile}
echo ""
# We need to have a sematext-agent pod running on each node
logging_proj=$(oc get projects| grep logging| awk '{print $1}'  | tail -1)
echo "==== Checking ${logging_proj} ==="
oc get pods -n ${logging_proj} -o wide > ${tmpfile}
for node in $(ansible ${first_master} -i ${invfile} -m shell -a "oc get
nodes --no-headers" --vault-password-file=${ansible_vault} | grep -v SUCCESS
| awk '{print $1}')
  do
    podfnd=$(grep ${node} ${tmpfile} | grep sematext-agent | grep "1/1")
    if [[ "${podfnd}" == "" ]] ; then
      echo "[ NOK ] pod sematext-agent is not running 1/1 on node ${node}"
      docker_pod_id=$(ansible ${node} -i ${invfile} -m shell -a "docker ps -
```

```
a | grep sematext-agent-docker" --vault-password-file=${ansible_vault} |
grep -v SUCCESS | grep -v FAILED | grep -v non-zero | awk '{print $1}')
      if [[ "${docker_pod_id}" != "" ]] ; then
        ansible ${node} -i ${invfile} -m shell -a "docker logs
${docker_pod_id}" --vault-password-file=${ansible_vault}
      else
        echo "no 'sematext-agent deployed'"
      fi
    else
      echo "[ OK ] pod sematext-agent is running 1/1 on node ${node}"
    fi
done
echo ""
#for node in $(grep logging-es ${tmpfile}|awk '{print $7}')
#  do
#    pod_st_1=$(grep ${node} ${tmpfile} | grep logging-es| awk '{print $2}'|
cut -d"/" -f1)
#    pod_st_2=$(grep ${node} ${tmpfile} | grep logging-es| awk '{print $2}'|
cut -d"/" -f2)
#    if [ "${pod_st_1}" != "${pod_st_2}" ] ; then
#      echo "[ NOK ] pod logging-es is not running as desired
${pod_st_1}/${pod_st_2} on node ${node}"
#    else
#      echo "[ OK ] pod logging-es is running as desired
${pod_st_1}/${pod_st_2} on node ${node}"
#    fi
#done
#echo ""
#for pds in curator kibana
#  do
#    pod_curator_ste_1=$(grep ${pds} ${tmpfile}| awk '{print $2}'| cut -d"/"
-f1)
#    pod_curator_ste_2=$(grep ${pds} ${tmpfile}| awk '{print $2}'| cut -d"/"
-f1)
#    pod_curator_node=$(grep ${pds} ${tmpfile}| awk '{print $7}')
#    if [ "${pod_curator_ste_1}" != "${pod_curator_ste_2}" ] ; then
#      echo "[ NOK ] pod ${pds} is not running as desired
${pod_curator_ste_1}/${pod_curator_ste_2} on node ${pod_curator_node}"
#    else
#      echo "[ OK ] pod ${pds} is running as desired
${pod_curator_ste_1}/${pod_curator_ste_2} on node ${pod_curator_node}"
#    fi
#done
rm -f ${tmpfile}
echo ""
#
echo "===== Checking infra pods on each node, openshift-node and openshift-
sdn ===="
# on each node we need to have a logging, sync, ovs and snd pod running
# the logging pods have already been tested, so one may test only
```

```
# the projects openshift-node and openshift-sdn
# we do have one pod in openshift-node but two pods per node in openshift-
sdn
for prj in openshift-node openshift-sdn
  do
    echo ""
    echo "==== Checking infra project ${prj} ==="
    oc get pods -n ${prj} -o wide > ${tmpfile}
    for node in $(ansible ${first_master} -i ${invfile} -m shell -a "oc get
nodes --no-headers" --vault-password-file=${ansible_vault} | awk '{print
$1}')
      do
        pod_name=$(grep ${node} ${tmpfile} | awk '{print $1}')
        for pdn in ${pod_name}
          do
            pod_st_1=$(grep ${node} ${tmpfile}| grep ${pdn} | awk '{print
$2}'| cut -d"/" -f1)
            pod_st_2=$(grep ${node} ${tmpfile}| grep ${pdn} | awk '{print
$2}'| cut -d"/" -f2)
            if [[ "${pod_st_1}" != "${pod_st_2}" ]] ; then
              echo "[ NOK ] pod ${pdn} is not running as desired
${pod_st_1}/${pod_st_2} on node ${node}"
            else
              echo "[ OK ] pod ${pdn} is running as desired
${pod_st_1}/${pod_st_2} on node ${node}"
            fi
        done
    done
    rm -f ${tmpfile}
done
echo ""
# searching for errors on masters messages log
echo "==== Checking log errors on masters ==="
# we need to check the syslog on each master for error messages
mymonth=$(date +%b)
myday=$(date +%d)
for master_node in $(ansible ${first_master} -i ${invfile} -m shell -a "oc
get nodes | grep master" --vault-password-file=${ansible_vault} | grep -v
SUCCESS | awk '{print $1}')
  do
    ansible ${master_node} -i ${invfile} -m shell -a "grep error
/var/log/messages | tail -n 10| grep ${mymonth} | grep ${myday} | grep -v
ansible-command" --vault-password-file=${ansible_vault} > ${tmpfile}
    msg_fnd=$(grep error ${tmpfile})
    if [[ "${msg_fnd}" != "" ]] ; then
      echo "[ NOK ] there are error messages in syslog in master
${master_node}, please review"
      ansible ${master_node} -i ${invfile} -m shell -a "grep error
/var/log/messages | tail -n 10| grep ${mymonth} | grep ${myday} | grep -v
ansible-command" --vault-password-file=${ansible_vault}
    else
```

```
      echo "[ OK ] no problems on master node ${master_node} found"
    fi
done
echo""
echo "==== Check for recent problems in systemd service logs ==="
cmd_status=$(ansible ${first_master} -i ${invfile} -m shell -a "oc adm
diagnostics analyzelogs" --vault-password-file=${ansible_vault} | grep
'Completed' | grep -v 'no errors')
if [[ "${cmd_status}" != "" ]] ; then
  echo "[ NOK ] one of the following services have a problem: ${cmd_status}"
else
  echo "[ OK ] The service 'atomic-openshift-node' and 'docker' are working
fine"
fi
echo ""
# testing nfs cluster storage
echo "==== Checking cluster persistent storage ==="
# we need to check the syslog on each node for refused messages
for node in $(ansible ${first_master} -i ${invfile} -m shell -a "oc get
nodes --no-headers" --vault-password-file=${ansible_vault} | grep -v SUCCESS
| awk '{print $1}' | grep -v master)
  do
    ansible ${node} -i ${invfile} -m shell -a "grep refused
/var/log/messages | tail -n 10| grep ${mymonth} | grep ${myday} | grep -v
ansible-command | grep -v 'dial tcp'" --vault-password-file=${ansible_vault}
> ${tmpfile}
    msg_fnd=$(grep refused ${tmpfile})
    if [[ "${msg_fnd}" != "" ]] ; then
      echo "[ NOK ] there are storage refused messages in syslog on node
${node}"
      echo "${msg_fnd}"
    else
      echo "[ OK ] no storage connection problems on node ${node} found"
    fi
    ansible ${node} -i ${invfile} -m shell -a "systemctl is-active
rpcbind.service" --vault-password-file=${ansible_vault} > ${tmpfile}
    rpcbind_stat=$(grep -v active ${tmpfile} | grep -v SUCCESS)
    if [[ "${rpcbind_stat}" != "" ]] ; then
      echo "[ NOK ] service rpcbind is not active on node: ${node}"
      ansible ${node} -i ${invfile} -m shell -a "systemctl status
rpcbind.service" --vault-password-file=${ansible_vault}
      fixable_errors=true
      rpcbind_errors=true
      echo "-------------------------------------------------------------"
    else
      echo "[ OK ] service rpcbind is active and running - ${node}"
      echo "-------------------------------------------------------------"
    fi
done
echo ""
```

```
#
echo "==== Checking openshift-web-console ==="
oc get pods --no-headers -n openshift-web-console -o wide | grep -vi
completed > ${tmpfile}
for pde in $(cat ${tmpfile} | awk '{print $1}')
  do
    pod_st_1=$(grep ${pde} ${tmpfile}| awk '{print $2}'| cut -d"/" -f1)
    pod_st_2=$(grep ${pde} ${tmpfile}| awk '{print $2}'| cut -d"/" -f2)
    pod_node=$(grep ${pde} ${tmpfile}| awk '{print $7}')
    if [[ "${pod_st_1}" != "${pod_st_2}" ]] ; then
      echo "[ NOK ] pod ${pde} is not running as desired
${pod_st_1}/${pod_st_2} on node ${pod_node}"
    else
      echo "[ OK ] pod ${pde} is running as desired ${pod_st_1}/${pod_st_2}
on node ${pod_node}"
    fi
done
echo ""
#echo "==== Create an application and test that it deploys correctly ==="
# TODO
# Step 1 - create new test namespace:
#ansible ${first_master} -i ${invfile} -m shell -a "oc create namespace ose-
platform-test" --vault-password-file=${ansible_vault} | grep -v SUCCESS
# Step 2 - add new test app
#ansible ${first_master} -i ${invfile} -m shell -a "oc new-app --
template=deploy-image -p DEPLOYMENT=ose-plattform-test -p
IMAGE_NAME=marthaler/demoapp -p NETWORK_PORT=8080 -n ose-platform-test" --
vault-password-file=${ansible_vault} | grep -v SUCCESS

## Check here if Pod is deployed correctly.

# Expose service
#ansible ${first_master} -i ${invfile} -m shell -a "oc expose svc/ose-
plattform-test -n ose-platform-test" --vault-password-file=${ansible_vault}
| grep -v SUCCESS

#app_route=$(ansible ${first_master} -i ${invfile} -m shell -a "oc get route
ose-plattform-test --no-headers -n ose-platform-test" --vault-password-
file=${ansible_vault} | grep -v SUCCESS | awk '{print $2}')

#diag_result=$(cat ${tmpfile} | grep Completed | grep "no errors")
#if [[ "${diag_result}" != "" ]] ; then
#  echo "[ OK ] application test deploy was sucessfuly"
#else
#  echo "[ NOK ] diagnositcs application test deploy problem, please review"
#  cat ${tmpfile}
#fi
#else
#  echo "sorry no internet access from cluster. - can't deploy test-image
from redhat"
#fi
```

```
#echo ""
#echo "==== Check the integrated heapster metrics can be reached via the API
proxy ==="
#ansible ${first_master} -i ${invfile} -m shell -a "oc adm diagnostics
metricsapiproxy" --vault-password-file=${ansible_vault} > ${tmpfile} 2>&1
#diag_result=`cat ${tmpfile} | grep Completed | grep "no errors"`
#if [ "${diag_result}" != "" ] ; then
#  echo "[ OK ] diagnositcs MetricsApiProxy"
#else
#  echo "[ NOK ] diagnositcs MetricsApiProxy problem, please review"
#fi
#
#--------------------------------------------------------------------------
---------
#                            OPTIONAL MAINTENANCE TASKS:
#--------------------------------------------------------------------------
---------
echo
"#########################################################################
############################"
echo ""
echo "===== Maintenance Tasks ===="
echo ""
echo
"#########################################################################
############################"
echo ""
if [[ "${fixable_errors}" ]] ; then
#
  if [[ "${pod_errors}" ]] ; then
    echo "==== Delete failed and crashed pods? ==="
    echo "y/n:"
    read delete_pods_now
    if [[ "${delete_pods_now}" == "y" ]] ; then
      oc get pods --no-headers --all-namespaces | grep -v Running | grep -v
Completed > ${tmpfile}
      for pod_name in $(cat ${tmpfile} | awk '{print $2}')
        do
          pod_namespace=$(grep ${pod_name} ${tmpfile}| awk '{print $1}')
          oc delete pod $pod_name -n $pod_namespace
        done
    fi
  fi
#
  if [[ "${rpcbind_errors}" ]] ; then
    echo "==== Try to restart all failed rpcbind services? ==="
    echo "y/n:"
    read restart_rpcbind_now
    if [[ "${restart_rpcbind_now}" == "y" ]] ; then
      for node in $(ansible ${first_master} -i ${invfile} -m shell -a "oc
```

```
get nodes --no-headers" --vault-password-file=${ansible_vault} | grep -v
SUCCESS | awk '{print $1}' | grep -v master)
        do
        rpcbind_node_status=$(ansible ${node} -i ${invfile} -m shell -a
"systemctl is-active rpcbind.service" --vault-password-file=${ansible_vault}
| grep -v SUCCESS)
        if [[ $rpcbind_node_status != "active" ]] ; then
          ansible ${node} -i ${invfile} -m shell -a "systemctl restart
rpcbind.service && systemctl enable rpcbind.service" --vault-password-
file=${ansible_vault}
          ansible ${node} -i ${invfile} -m shell -a "systemctl status
rpcbind.service" --vault-password-file=${ansible_vault}
        fi
      done
    fi
  fi
#
else
  echo "notihing to do.. Everyting is all right. ;)"
fi
```

Last update: **2019/09/04 12:59**