**My Project**

**Final Project Definition Template**

By:

Student 1 Yoav Zinger

Student 2 Arad Gonen

Student 3 Mariel Borodkin

z

Yaniv

Git:

[aradgonen/OpenShift-Sentinel: Cyber Course Project - SIEM (github.com)](https://github.com/aradgonen/OpenShift-Sentinel)

Table of content

[1. Project description 2](#_Toc101172187)

[2. Related Work 2](#_Toc101172188)

[3. Functional Description / Requirements 3](#_Toc101172189)

[4. Architecture 3](#_Toc101172190)

[4.1. Each Module description 3](#_Toc101172191)

[4.1. Terms 4](#_Toc101172192)

[5. Work plan 4](#_Toc101172193)

[6. Openshift 5](#_Toc101172194)

[6.1. IPI mode 5](#_Toc101172195)

[6.2. Create new VM 5](#_Toc101172196)

[6.3. SSH hardening for secure passwordless access 6](#_Toc101172197)

[6.3.1. Set a custom SSH port 6](#_Toc101172198)

[6.3.1.1. Backup configuration file 6](#_Toc101172199)

[6.3.1.2 Associate PIP 6](#_Toc101172200)

[6.3.1.3 Add network rule 6](#_Toc101172201)

[6.3.1.3.1 Azure port rule 7](#_Toc101172202)

[6.3.1.3.2 Linux machine 7](#_Toc101172203)

[6.3.2. Set additional parameters in sshd\_config 7](#_Toc101172204)

[6.3.3. VM reset password 8](#_Toc101172205)

[6.4. Cluster Deployment 8](#_Toc101172206)

[6.4.1. IPI mode deployment 9](#_Toc101172207)

[6.4.2. Do if getting 'ClientID expired' error 9](#_Toc101172208)

[6.4.3. Now small title 9](#_Toc101172209)

[6.4.4. Now small title 9](#_Toc101172210)

[6.5. Deploy application from github 9](#_Toc101172211)

[6.6. Cluster Logging 10](#_Toc101172212)

[6.6.1. Update Cluster version to 4.10 10](#_Toc101172213)

[6.6.2. Install Elasticseach operator 11](#_Toc101172214)

[6.6.3. Create an Openshift Logging instance 11](#_Toc101172215)

[6.6.4. Create an Openshift Logging instance 12](#_Toc101172216)

[6.6.5. Create an Openshift Logging instance 12](#_Toc101172217)

[6.7. Policy Deployment 12](#_Toc101172218)

[6.7.1. Red Hat Advanced Cluster Management for Kubernetes 12](#_Toc101172219)

[6.7.1.1. Installation 13](#_Toc101172220)

[6.7.2. OPA Gateway 13](#_Toc101172221)

[6.8. Some tite 13](#_Toc101172222)

[7. Client side 13](#_Toc101172223)

[7.1. Usage Illustration 13](#_Toc101172224)

[7.2. Mockup 14](#_Toc101172225)

[8. Server Side 16](#_Toc101172226)

[8.1. API 16](#_Toc101172227)

# Project description

Tool for the compliance framework. The tool is used to deploy compliance policies onto OpenShift clusters and monitor the policies results. The tool is used by compliance officers in organizations to enforce and visualize compliance status on OpenShift clusters.

# Related Work

Our work related to an Open Source project which eventually supposed to join Red Hat's official Open Source repo:

<https://github.com/openshift-4-compliance/openshift-4-compliance-automation>

Our project is supposed to deliver a deployment strategy for our tool, including automatic connector to cluster's logs.

* **Sync Controller** - Used to sync policies from a git repository with the UI. A user can choose to deploy policies from the specified Git repository (as described in the diagram).
* **Frontend** - Allows the user to deploy policies via a wizard that syncs with the sync controller for policy deployment.
* **Policy Controller** - Used to receive data from the user & frontend, and deploy the policies via the specified tool
  + Red Hat Advanced Cluster Management for Kubernetes

Open Policy Agent Gatekeeper

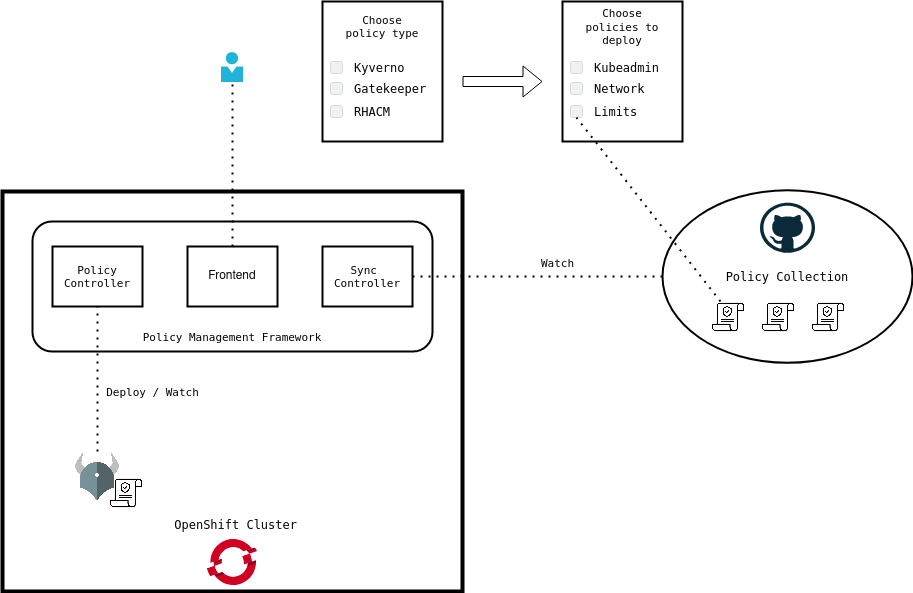
Many of strategy and design is based on Red Hat's documentation:

* <https://docs.openshift.com/>
  + [2. Installing Logging | Logging | OpenShift Container Platform 4.9](https://docs.openshift.com/container-platform/4.9/logging/cluster-logging-deploying.html#cluster-logging-deploy-console_cluster-logging-deploying)

# Functional Description / Requirements

* **Collect data at scale and detect** based on Openshift audit and security logs for **security information event management** (SIEM) solution.
* **Investigate and respond to incidents rapidly**– Provide **security orchestration automated response** (SOAR) solution of specified tasks.

# Architecture



## Each Module description

In this sub section you will describe the responsibility of each module… in case the module is a combination of several sub modules you should drill down and describe each of this sub modules as well.

## Terms

The following terms has been used in this document

|  |  |
| --- | --- |
| OCP | Openshift Container Platform |
| IPI | Installer-provisioned installation |
| PIP | Public IP |
| VM | Virtual Machine |
| EFK | Elasticsearch, Fluentd, and Kibana |
| RHACM | Red Hat Advanced Cluster Management for Kubernetes |
| OPA | Open Policy Agent |
|  |  |

# Work plan

This section will present the work plan of the project, the work plan must include milestone and responsibility of team members

Mariel Borodkin – Overall cybersecurity design and planning, DevOps

Arad Gonan – SIEM FE, SOAR functionality

Yoav – BE, custom policies

29/10/21 – Choose idea, meet with RedHat for collaboration

14/11/21 – Idea and work agreement established for Open Source (OSS) project.

14/11/21 - Project Idea was approved by supervisor.

28/11/21 - Deploy Dev environment for research test purposes of the technology.

14/12/22 – POC succeed.

28/12/22 – First security custom policy deployed for monitoring on Dev environment.

07/01/22 – HLD design architecture, plan components, choose technology solutions.

14/02/22 – Basic client - server side for visualization of OpenShift logs (SIEM)

14/02/22 – Basic client - server side for visualization of OpenShift logs (SIEM)

21/02/22 – Basic SOAR functions resolving policy security threats.

01/03/22 – Own custom policy for Database authentication security.

10/03/22 – Custom PatternFly components is ready to used.

13/03/22 – Authenticate to app via SSO (Optional to client's preference)

16/03/22 – Add login functionality to SIEM using RedHat SSO.

21/03/22 – Admin page and management pane to SIEM.

05/04/22 – Basic GraphQL schemes is functioning and UI in pooling using GraphQL/Appolo client.

05/04/22 – ElasticSearch instance is up and used for OpenShift audit log forwarding.

07/04/22 – Frontend is ready to visualize two types of graphs.

21/04/22 – SIEM is done – both backend and frontend.

01/05/22 – Basic SOAR functionality is deployed in Sentinel backend.

08/05/22 – All OpenShift-Compliance policies is deployed on the Dev cluster.

10/05/22 – SIEM is monitoring all compliance policies.

# Openshift

This section includes all Openshift environment related information,

## IPI mode

Openshift will be provisioned on Microsoft's cloud technology 'Azure'.

We deployed OCP in IPI mode

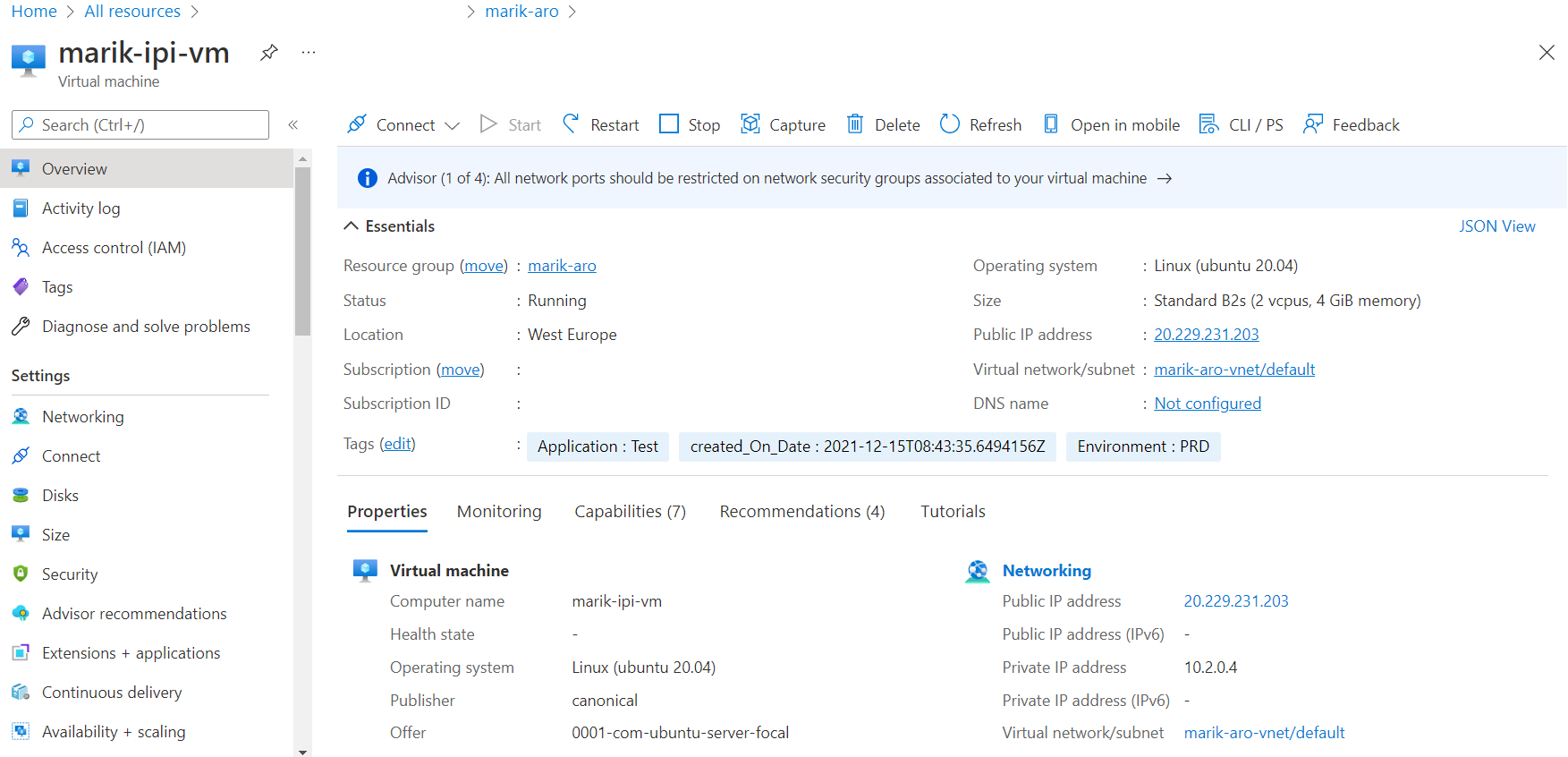
It's recommending to read this documentation in addition to our file before continuing. For more information about IPI mode check the following links:

[4. Installing a cluster quickly on Azure - Installing on Azure | Installing | OpenShift Container Platform 4.6](https://docs.openshift.com/container-platform/4.6/installing/installing_azure/installing-azure-default.html#installation-launching-installer_installing-azure-default)

## Create new VM

As noted in the previous section [IPI mode](#_IPI_mode), you need to create VM to provision your OCP. For that, you need to create new VM in Azure.

You need to create a Linux (ubuntu 20.04) machine.



For this you can follow those instructions: [7. Tutorial - Create and manage Linux VMs with the Azure CLI - Azure Virtual Machines | Microsoft Docs](https://docs.microsoft.com/en-us/azure/virtual-machines/linux/tutorial-manage-vm)

## SSH hardening for secure passwordless access

The goal is mitigating SSH based attacks – Top Best SSH Security Practices that we implemented in Azure environment and should be implemented in each RHEL server with SSH access.

The following steps turn your SSH server into a rock-solid communication daemon.  
Note: most of these tips will require you to restart the SSH service to apply the changes. You can do that by issuing:  
service sshd restart

### Set a custom SSH port

This is one of the oldest and most popular ways to obscure the SSH service. Why? Because by default, SSH comes listening on port 22, which is widely known among attackers and security tools/port scanners that launch brute force attacks against it. While this is considered security by obscurity, it helps eliminate lots of noise on port 22. There are a few steps before continuing:

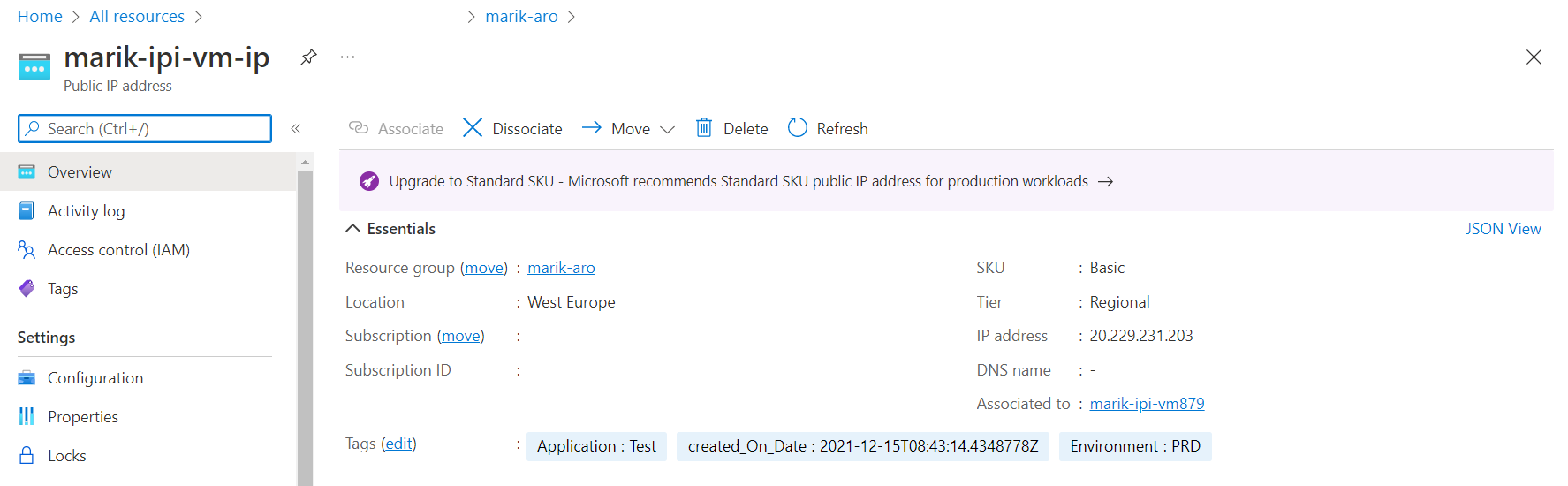
### Backup configuration file

First, backup the configuration file before making major changes.

# cp /etc/ssh/sshd\_config ~/sshd\_config\_original

### Associate PIP

Create new resource *Public IP* and associate it to the VM. Public IP addresses in Azure are used for public connections to Azure resources.



You can also add Alias to DNS if you want to access to server by name and not by IP.

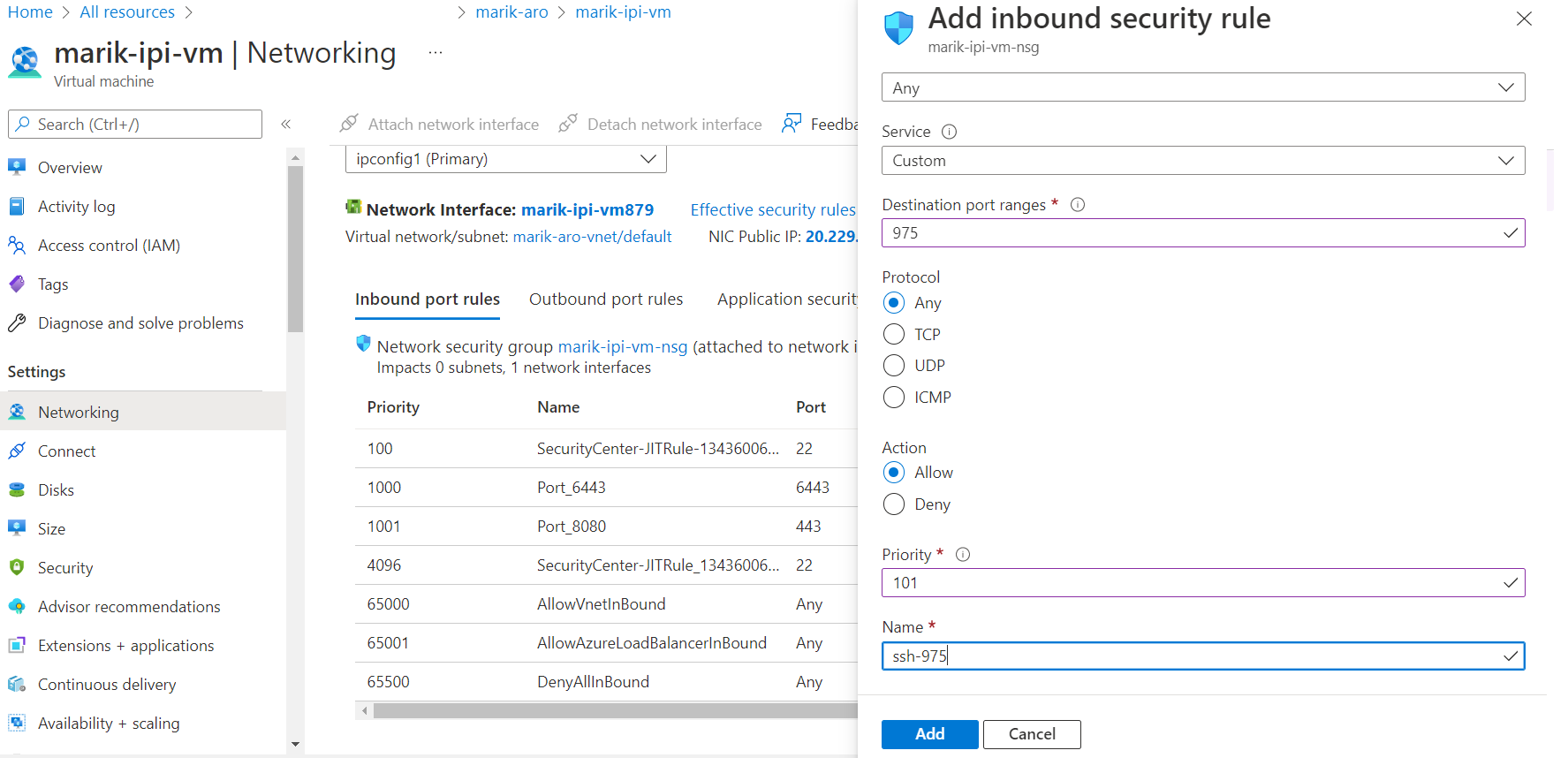
You can follow those instructions: [5. Quickstart: Create a public IP address - Azure portal - Azure Virtual Network | Microsoft Docs](https://docs.microsoft.com/en-us/azure/virtual-network/ip-services/create-public-ip-portal?tabs=option-1-create-public-ip-standard)

### Add network rule

Add no-default port for SSH-access, in our case: **975 TCP** for both VM in Azure and in ubuntu configuration:

### Azure port rule

Add inbound rule in VM > Networking > inbound port rules.



### Linux machine

You need to run the following commands:

Connect to VM

Run:

# firewall-cmd --add-port 975/tcp --permanent  
# firewall-cmd --reload  
# firewall-cmd --list-ports

Edit your [SSH main config file](https://www.ssh.com/ssh/sshd_config/):

vim /etc/ssh/sshd\_config

Search for: Port

Then set it to something different than 22, in our case 975, like:  
Port 975  
And save the file.  
After changing the port on a SELinux system, you have to tell SELinux about this change.  
# semanage port -a -t ssh\_port\_t -p tcp 975  
# service sshd restart

### Set additional parameters in sshd\_config

Additional parameters that we are going to change in the /etc/ssh/sshd\_config are:

* Disable Root Login
* SSH Passwordless Login
* Set Idle Timeout interval
* Disable Empty Passwords
* Disable X11 Forwarding
* Limit Max Authentication attempts

All those changes will be done in /etc/ssh/sshd\_config file by changing the values of the next parameters and saving the file:

PermitRootLogin no

MaxAuthTries 3

PubkeyAuthentication yes

PermitEmptyPasswords no

PasswordAuthentication no

X11Forwarding no

ClientAliveInterval 180

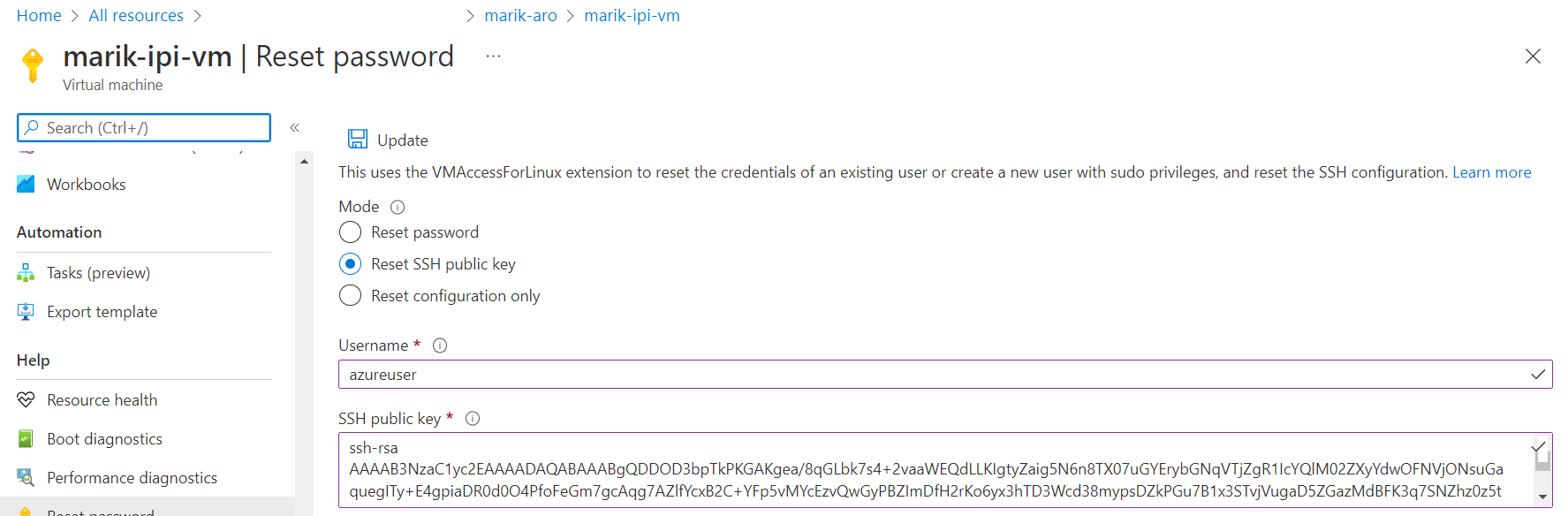
ClientAliveCountMax 3

# service sshd restart

### VM reset password

Go to properties of your VM → Reset Password

Change to Mode - reset SSH Public key, choose user azureuser (or some other name) and paste public key of your laptop / machine where from you want to connect to VM by SSH and click “Update”



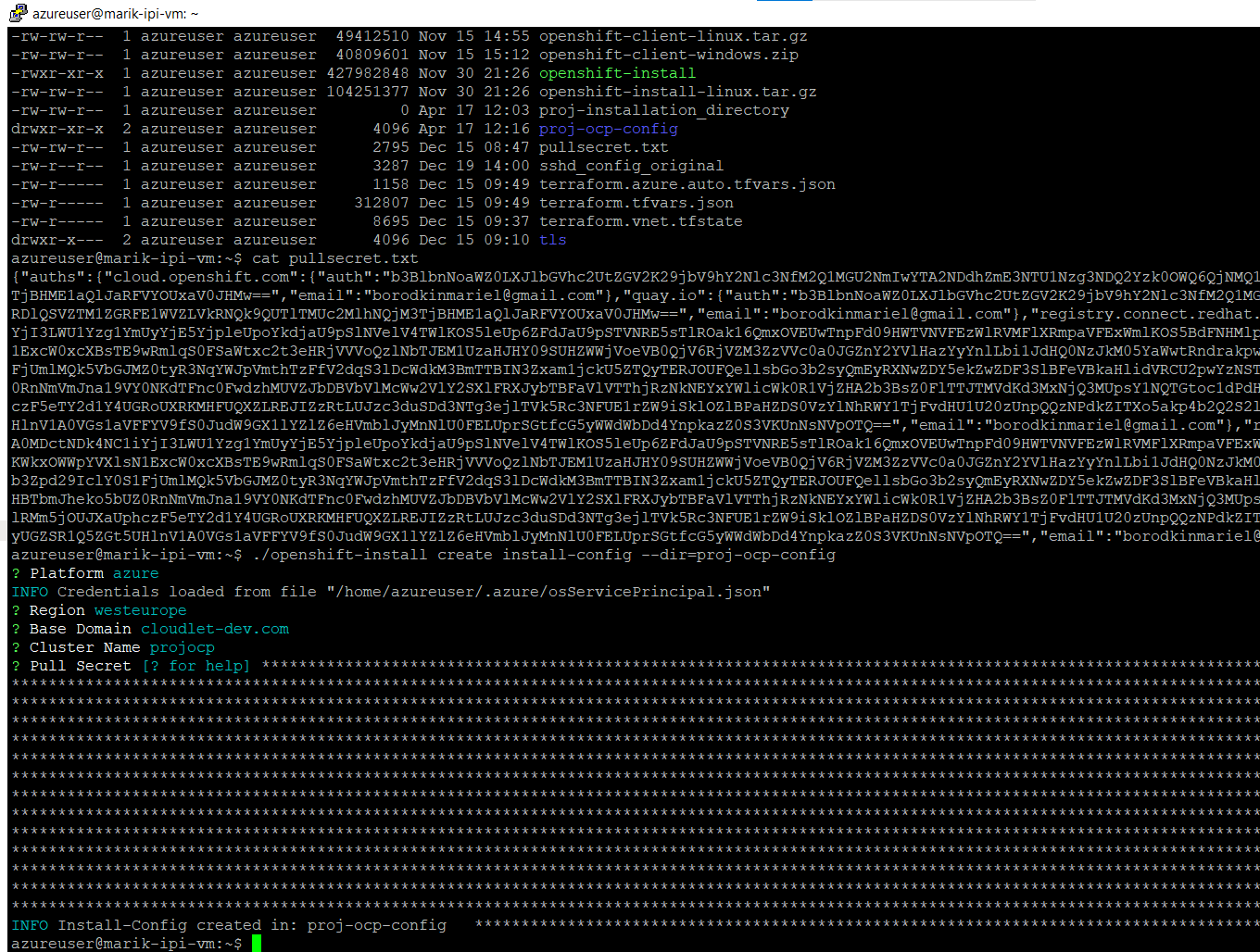
## Cluster Deployment

Complete to this section after you've done all preparations in [previous section](#_Set_a_custom).

From the installer VM deploy OCP Cluster in IPI mode according to the official Red Hat procedure installing-azure-default. In the deployment process you will require some information - you can find this info below

### IPI mode deployment

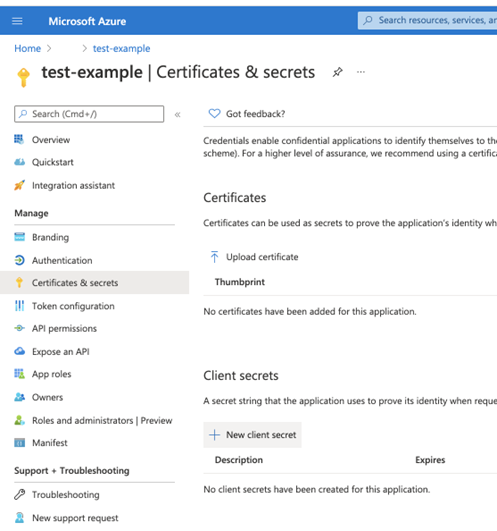
In this section, we are going to deploy the cluster according to [IPI mode](#_IPI_mode) section.



### Do if getting 'ClientID expired' error

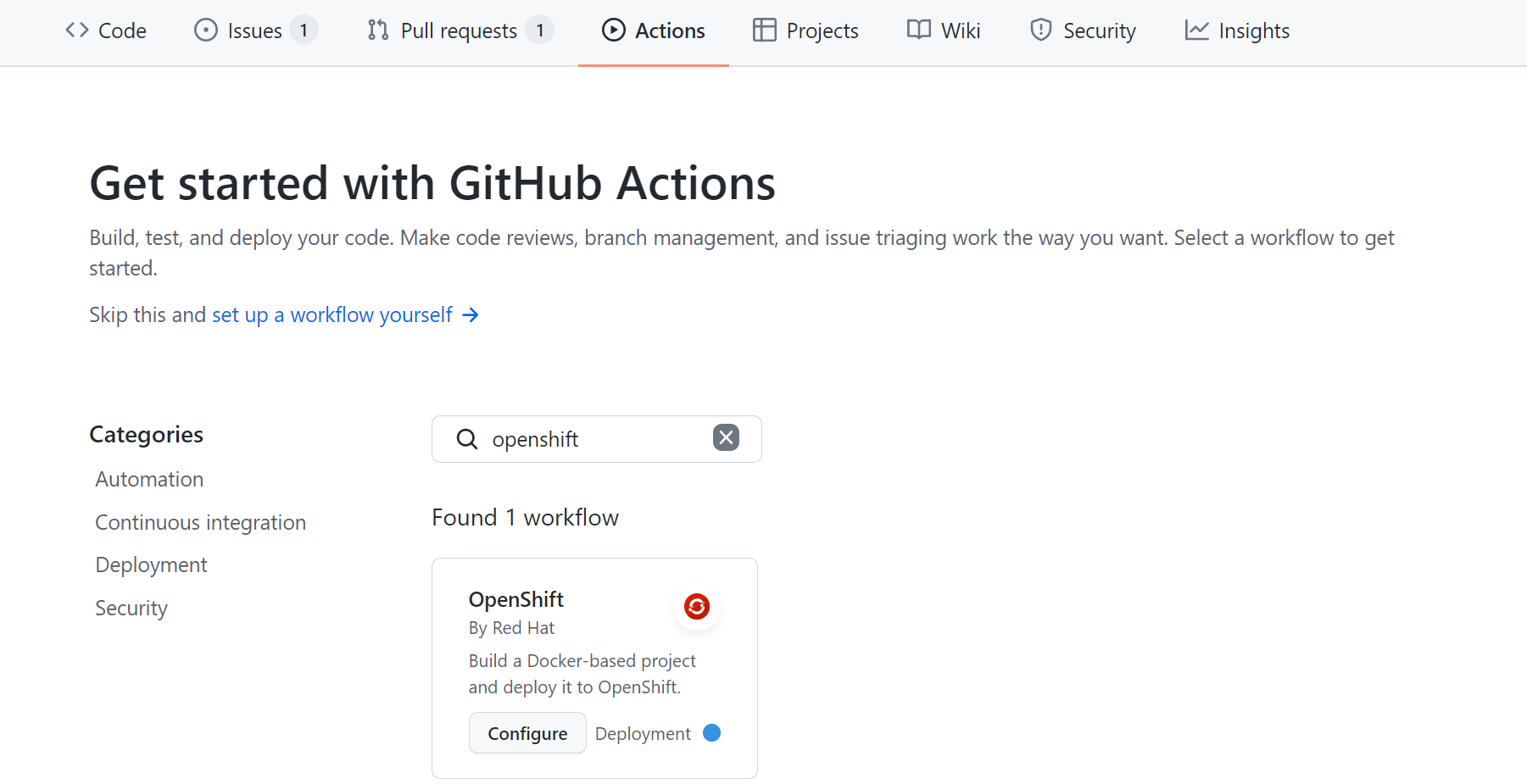
From time to time ClientID can expire. To overcome this issue you need to create a new ClientID and provide this clientID permissions on the Subscription level.

1. Register application with Application developer role
2. Go to Certificates and secrets and create new secret.
3. Copy this secret and client id for IPI installation.
4. Give permissions to app registration created to create resources in Azure



## Deploy application from github

Now when we have cluster we can deploy the application to the OCP via *Github Action* in the following way:

1. Create action
2. 

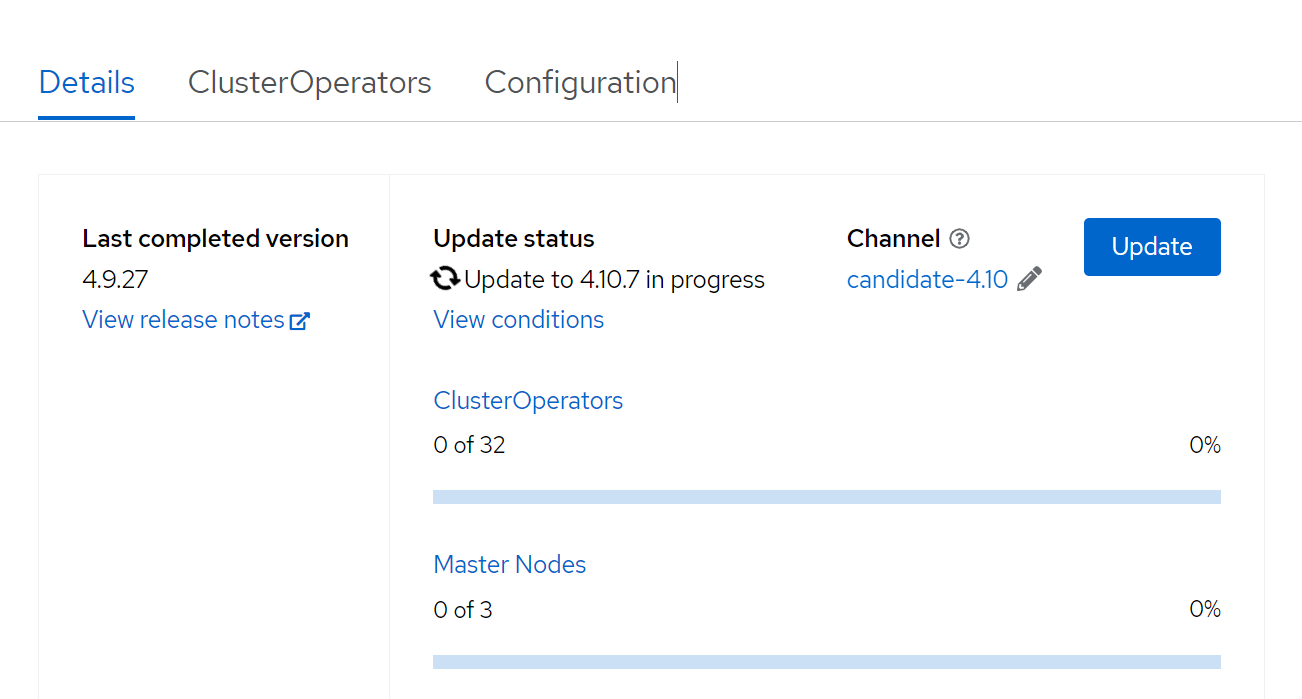
## Cluster Logging

For logging, we decided to go on open source solution EFK (Elasticsearch, Fluentd, and Kibana), for this we'll need to install Elastic and Cluster logging operators.

### Update Cluster version to 4.10

If your cluster version is 4.10.x you can continue to the next step.

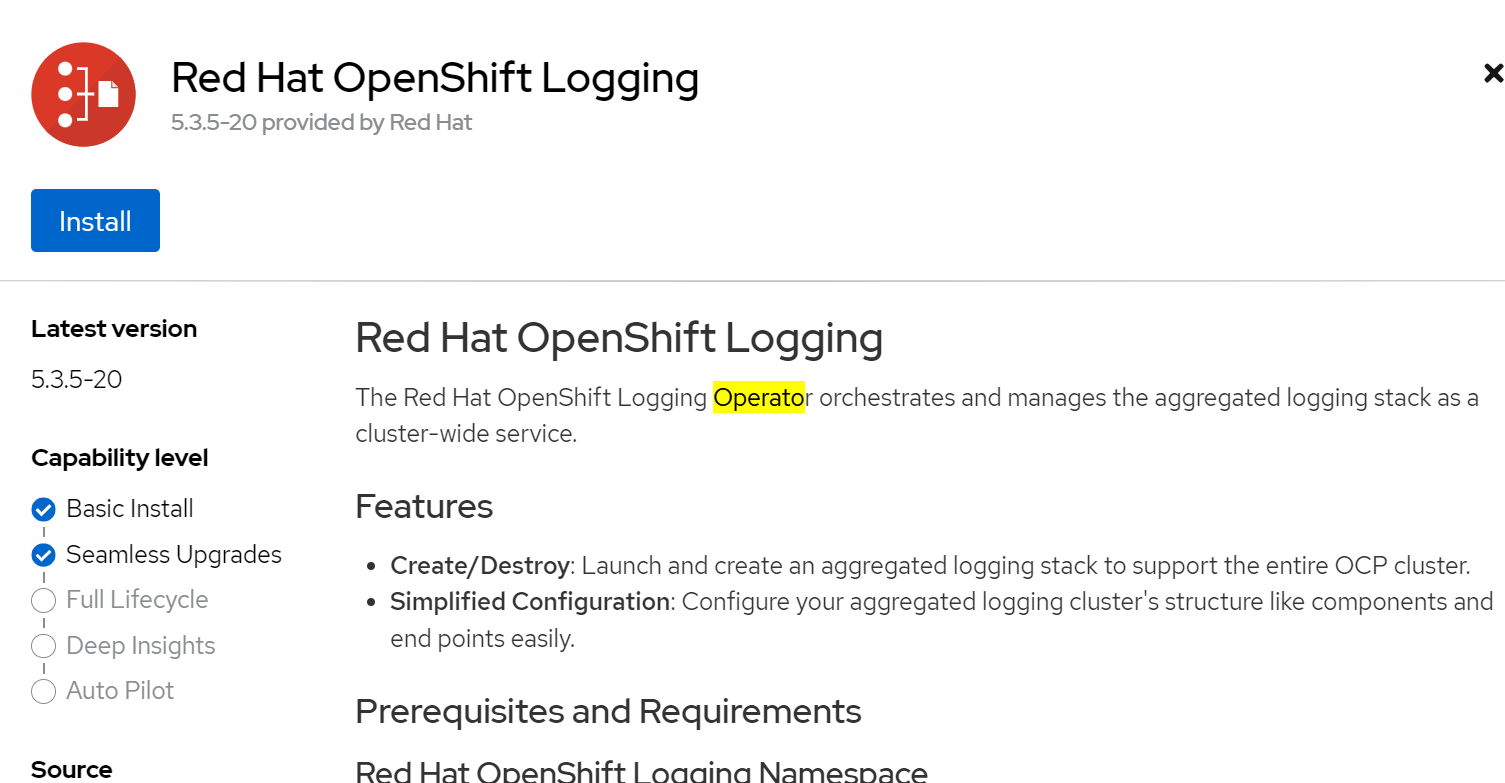
Otherwise, go to cluster settings, change channel to 4.10candidate and update to 4.10.x release.



### Install Elasticseach operator

We need to install elasticsearch blabla

### Create an Openshift Logging instance

Create 

After that, you create logging instance in Administration → Custom Resource Definitions with the following YAML:

apiVersion: "logging.openshift.io/v1"

kind: "ClusterLogging"

metadata:

  name: "instance"

  namespace: "openshift-logging"

spec:

  managementState: "Managed"

  logStore:

    type: "elasticsearch"

    retentionPolicy:

      application:

        maxAge: 1d

      infra:

        maxAge: 7d

      audit:

        maxAge: 7d

    elasticsearch:

      nodeCount: 3

      storage:

        storageClassName: "managed-premium"

        size: 200G

      resources:

          limits:

            memory: "5Gi"

          requests:

            memory: "5Gi"

      proxy:

        resources:

          limits:

            memory: 256Mi

          requests:

            memory: 256Mi

      redundancyPolicy: "SingleRedundancy"

  visualization:

    type: "kibana"

    kibana:

      replicas: 1

  collection:

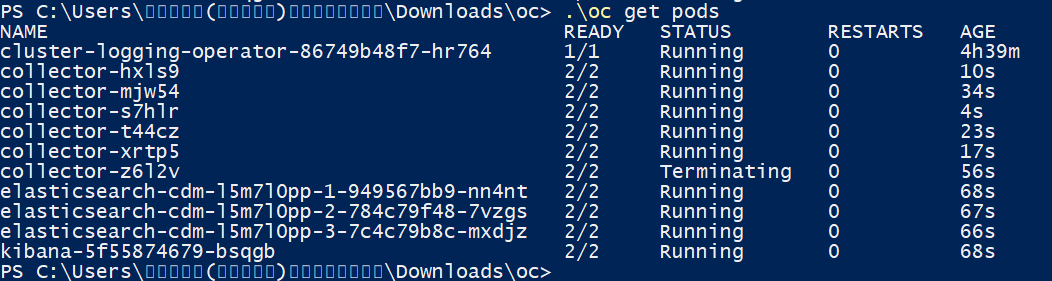
    logs:

      type: "fluentd"

      fluentd: {}

**Validate that your *StorageClusterName* is configured, if not configure with your resource provider.**

After deploying, this should be the status of the pods:



## Log Forwarding

## Policy Deployment

As we want to monitor policies and the cluster, we'll now deploy policies for example from the repository to my created environment. To achive this you need the following tools:

* [Red Hat Advanced Cluster Management for Kubernetes](https://github.com/open-cluster-management)
* [Open Policy Agent Gatekeeper](https://github.com/open-policy-agent/gatekeeper)

In the coming section, we'll explain about them and their uses.

### Red Hat Advanced Cluster Management for Kubernetes

Make sure you install RHACM on your cluster. Make sure you meet the requirements

[Chapter 1. Installing Red Hat Advanced Cluster Management for Kubernetes 2.0 | Red Hat Customer Portal](https://access.redhat.com/documentation/en-us/red_hat_advanced_cluster_management_for_kubernetes/2.0/html/install/installing)

### Installation

You need to create a new secret and use PullSecret created in [Cluster Deployment](#_Cluster_Deployment).

Run the following command:

oc create secret generic mariksec -n openshift-logging --from-file=.dockerconfigjson="C:\Users\מריאל(מאריק)בורודקין\Downloads\oc\pullsec.txt" --type=kubernetes.io/dockerconfigjson

Apply created file on operator:

OperatorGroup.yaml

apiVersion: operators.coreos.com/v1

kind: OperatorGroup

metadata:

name: loggingoperatorgroup

spec:

targetNamespaces:

- openshift-logging

oc apply -f OperatorGroup.yaml

Apply the subscription

Sub.yaml

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

name: acm-operator-subscription

spec:

sourceNamespace: openshift-marketplace

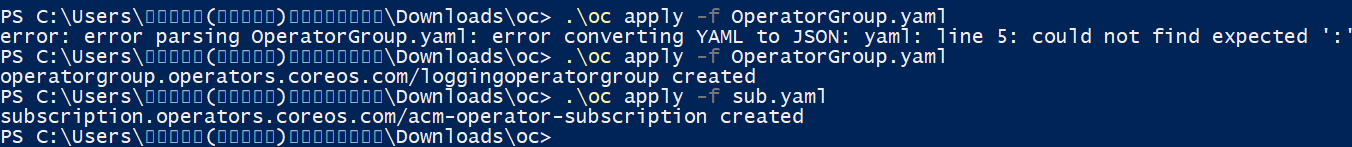
source: redhat-operators

channel: release-2.0

installPlanApproval: Automatic

name: advanced-cluster-management

oc apply -f sub.yaml



With RHACM, we will deploy policies. We can to it via GitOps, first that first run command to initiate:

oc patch clusterrolebinding.rbac open-cluster-management:subscription-admin -p '{"subjects": [{"apiGroup":"rbac.authorization.k8s.io", "kind":"User", "name":"<your-username>"}]}'

### OPA Gateway

Bla bla

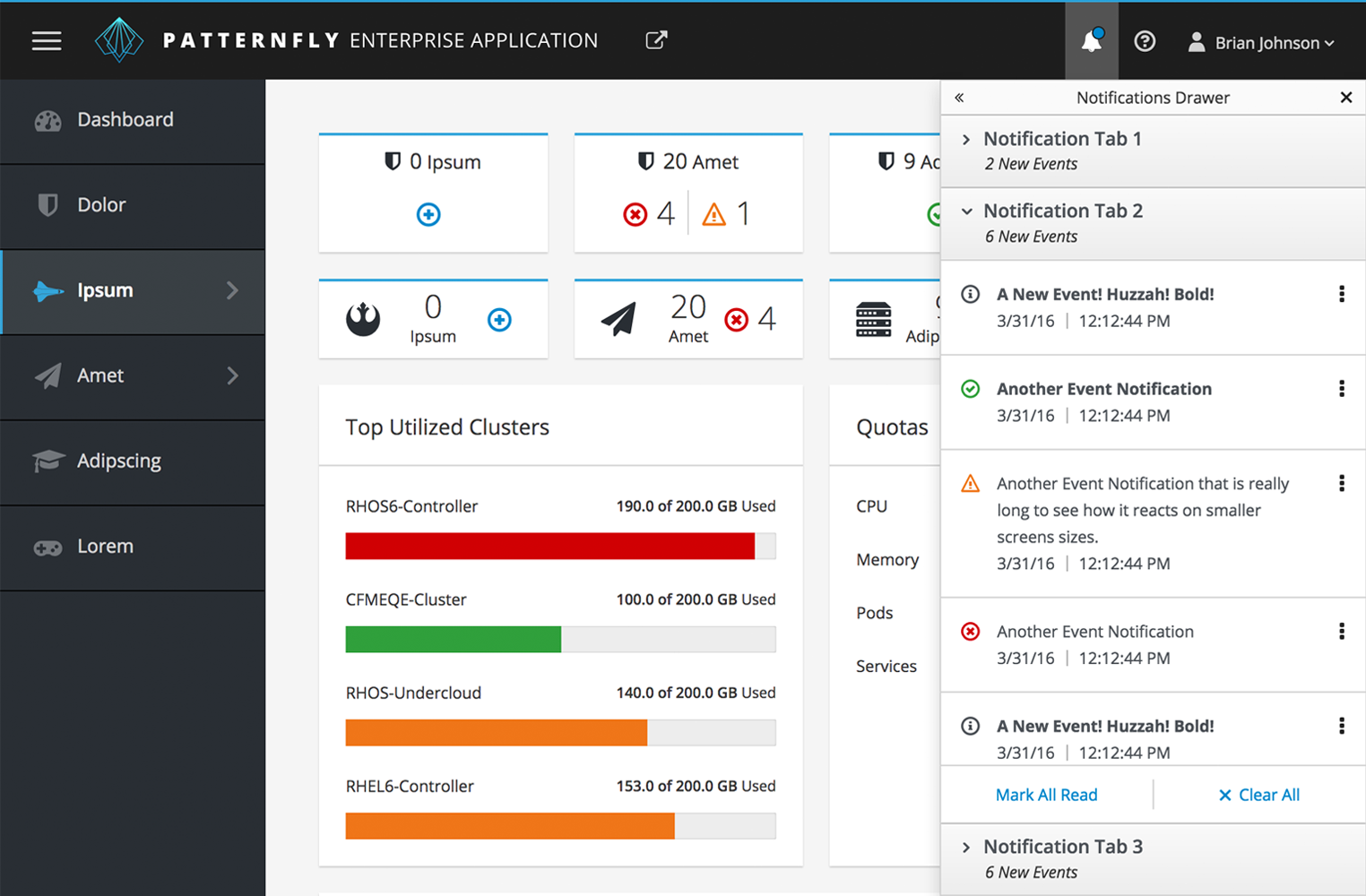
## Some tite

# Client side

In projects that include a client side – and app or website need to fill in this section.

This section describe the look and operation of the client UI:

Based on React, using patternfly UI framework. For example:



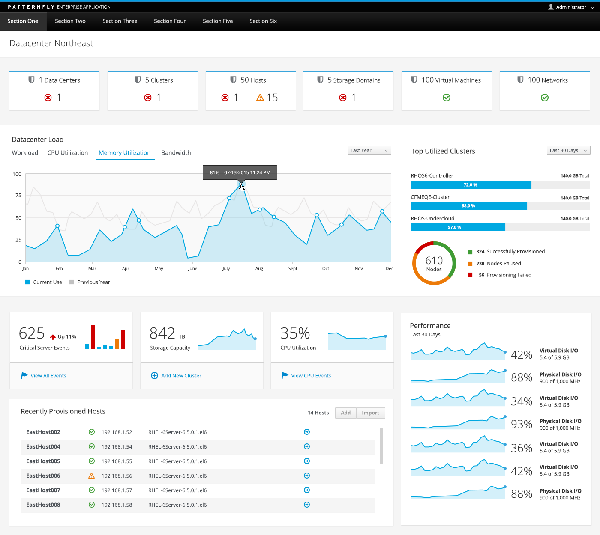
## Usage Illustration

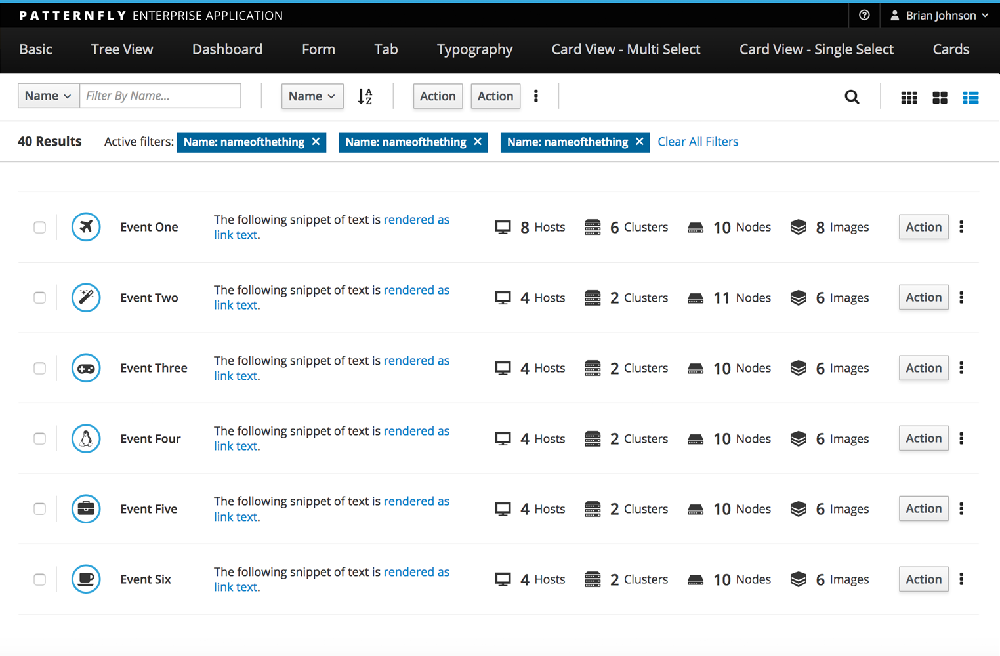
User will deploy our application with automation deployment strategy on a node in OpenShift environment. Policies will be deployed as YAML files:

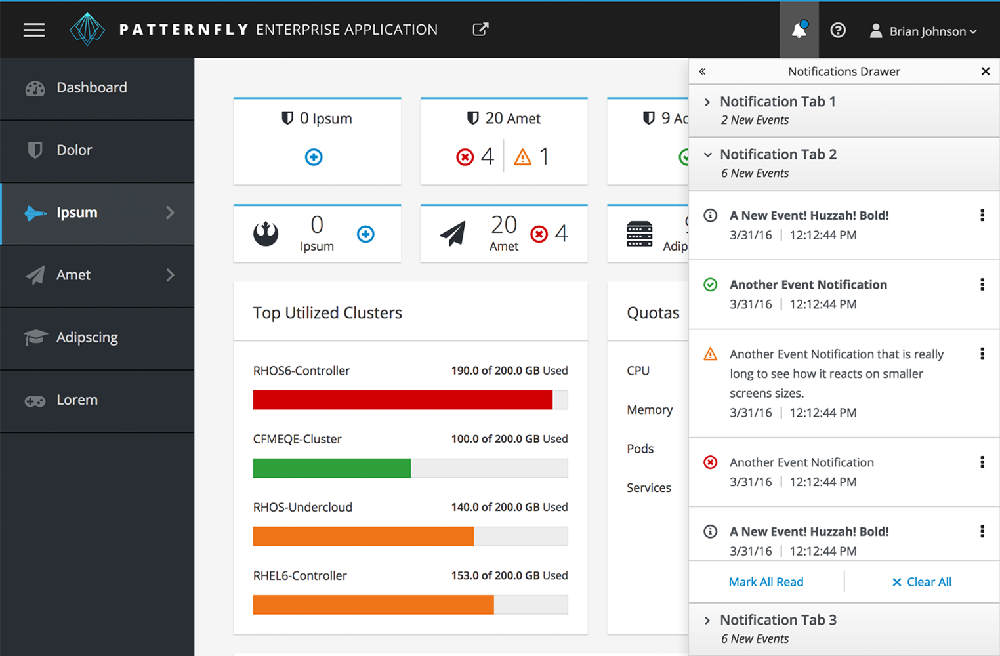
oc apply -f policy.yaml

## Mockup

This is an illustration, not our app:







Source: [PatternFly | open interface project](https://www.patternfly.org/v3/)

# Server Side

Based on NodeJS BE with MongoDB Database. Using Reddis as cache.

## API

Exposing API that checks for CVEs for policy exploits in exploit-DB

[Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers (exploit-db.com)](https://www.exploit-db.com/)

Before deploy a YAML.