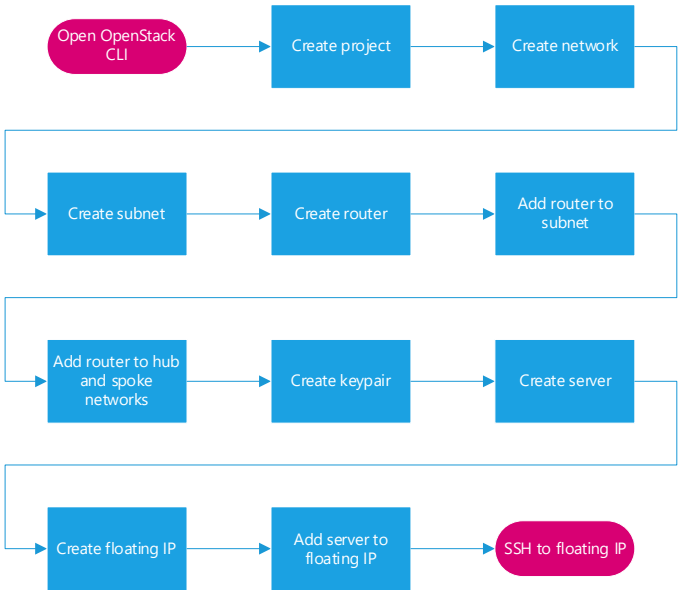
1.1 UC-0001 Provision a” clean” virtual machine on OpenStack

Use case ID	UC-0001
Use Case Name	Provision a” clean” virtual machine on OpenStack
Author/Owner	Valdemar Lemche / Valdemar Lemche
Summary	Must illustrate that it is possible to manage infrastructure as code by creating a fully software defined infrastructure stack, including connecting to the created virtual machine. • Create dependencies required to connect to the virtual machine over DSV network traffic. • Create a virtual machine in OpenStack • Connect to virtual machine from DSV jump server. Flow can be executed on a local workstation with the correct network access.
Actors	• DevOps Engineer
Preconditions	1. OpenStack environment is connected to DSV general network infrastructure 2. Service Principal with sufficient access to manage all resources in project. 3. Approved OS Image is available 4. SSH public key for the initial operator 5. Allocated DSV IP address
Flow	 **Figure 1:** UC-0001 flowchart giving a high-level description of process.

Commented [KAD1]: Do we dare to state ‘approved image’ ?

Commented [VLD2R1]: Tjo …

Commented [KAD3R1]: I think the safe choice will be to leave it out for the MVP demo/example

Commented [KAD4R1]:

Postconditions	Flow will have produced 1. A new project in OpenStack. 2. A spoke network we can connect our servers to. 3. A subnet in the network that defined the IP address range. 4. A router that will provide connectivity between the OpenStack Hub Network and the spoke network. 5. A generic SSH public key that we can use to login to the server as administrator. 6. A new virtual machine (server) that we can start deploying software to 7. A destination NAT and source NAT address that we can use to connect to the server with.
Level	Very high summary
Trigger	On demand when requested by coder
Stakeholders	• Program Architects • System Architects • OpenStack product teams leads

Table 1: UC-0001 Provision a ”clean” vrtuel machine on Openstack

1.2 UC-0002 Migrate an application (2 tier) from Legacy VMware to OpenStack

Use case ID	UC-0002
Use Case Name	Migrate an application (2 tier) from Legacy Vmware to OpenStack
Author/Owner	Valdemar Lemche / Valdemar Lemche
Summary	Migrate SonarQube environment from Core Inside on VMware to OpenStack while keeping existing database content, and minimizing configuration changes to application. Purpose is to enable SonarQube environment to be both horizontally and vertically scaled using OpenStack IaaS and automation, and moving to a OpenSource database implementation.
Actors	• DevOps Tool Chain Engineer • Platform Services Solution Architect • Network Operation Center Engineer

Preconditions	• OS image with ITO DevOps standard
	• Network access to DevOps Artifactory Docker registry
	• SSL certificates for new SonarQube servers
	• Approval for several days of downtime or additional licenses
	• Firewall openings to OpenStack Networks from
	  ○ From Bitbucket in Core Inside to OpenStack SonarQube VMs
	  ○ From Jenkins slaves to to OpenStack SonarQube VMs
	  ○ From Jenkins slaves to OpenStack APIs
	  ○ From ALM proxy to OpenStack SonarQube VMs
	• Possibility to order PostgreSQL (PGSQL) cluster from Platform Services on OpenStack

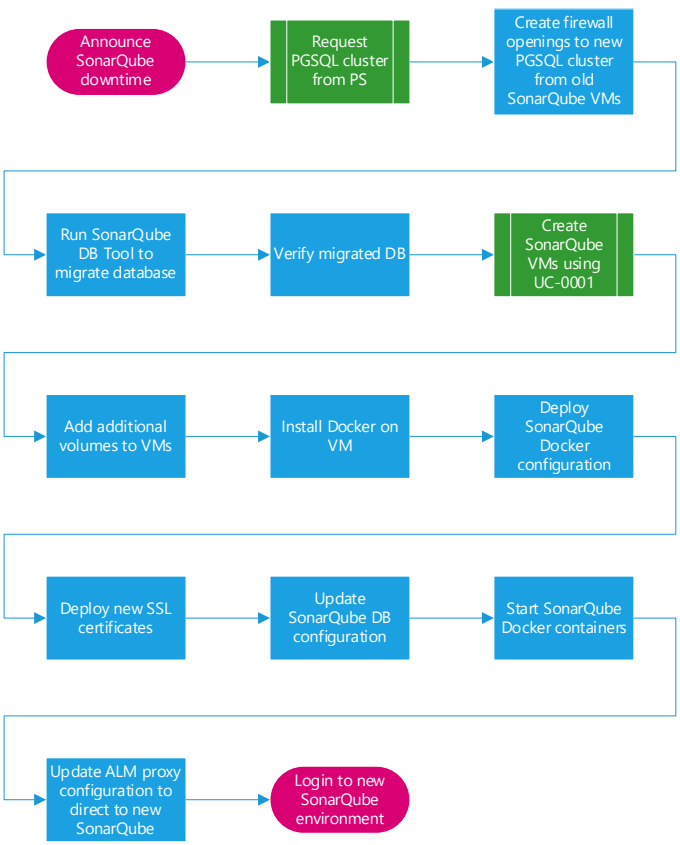
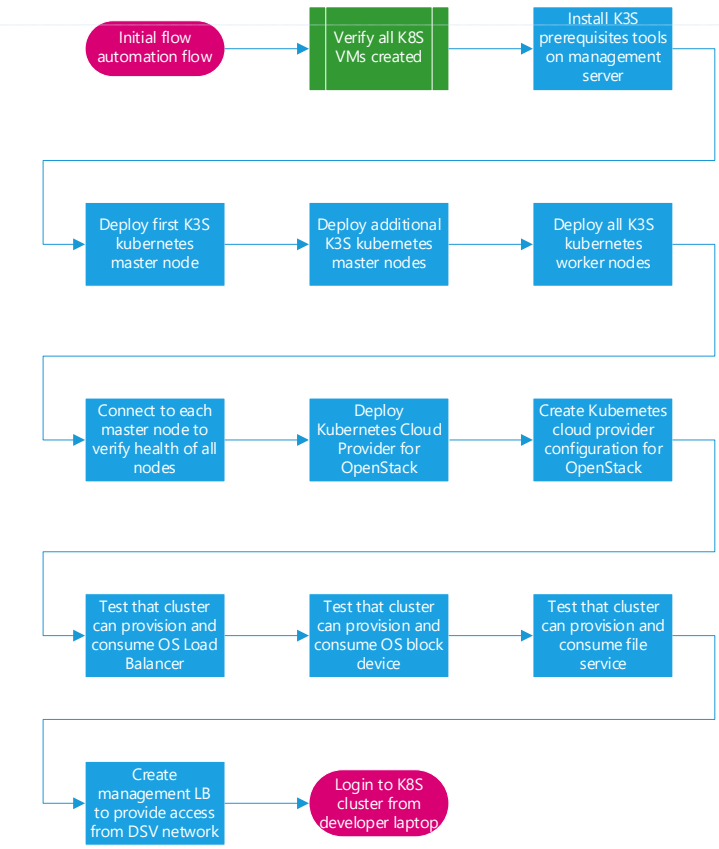
Flow	
Postconditions	• SonarQube Database have been migrated from Microsoft SQL Server to PostgreSQL in OpenStack • SonarQube application have been migrated to use new VMs and database with minimal changes. • SonarQube is still accessible using **existing URL** so existing users will not have to change how they consume the service.
Level	Summary
Trigger	• DevOps Engineer initiates flow as a part of sprint user story
Stakeholders	• Service Owner of SonarQube • All developers

Table 2: UC-0001 Migrate an application (2 tier) from Legacy Vmware to OpenStack

1.3 UC-0003 Build K8S cluster to run Private Cloud app

Use case ID	UC-0003
Use Case Name	Build K8S cluster to run Private Cloud app
Author/Owner	Kim Andersen – DSV, Ankur Jha – IBM / Valdemar Lemche
Summary	Purpose of this use case it to illustrate the Process of Building a k8s Cluster on top of Open stack, This Deployment is a Fully Automated Process which need to be run on a secured Channel either from DSV Workstation or From DSV Network or Bastion host • Create a Virtual Machine with inbuild image. • Create a Flavour on it. • Create a Networks on top of it, which includes Router & Public & Private Network • Create key pair to access it from Bastion host or from DSV networks • Create a template for whole cluster, it just so next time deployment will be faster • Run the deploy command by passing number of node count & other runtime variables Hop into the cluster with those key pair through defined accessible Network
Actors	• DevOps Engineer

Preconditions	• Extending on 'UC-0001 – Provision a" clean" Virtual machine on Open stack' to create the following virtual infrastructure components:
	  ○ A bastion host (jump host)
	  ○ A set of VMs for master nodes (typically 3)
	  ○ A set of VMs (between 1 and n) for worker nodes.
	• Choice of on-prem Kubernetes distribution determined. (Assumption is that K3S is used in MVP)
	• OpenStack Application Credentials to configure cloud provider in K8S

Flow	
Post conditions	Functioning K8S cluster where: • Kubernetes API can be accessed from 'general DSV network' – via Open stack Load balancer having its VIP on the Provider net. • Basic Kubernetes deployment can be created in cluster and scaled. E.g. Nginx web servicer Replica Set with n number of Pods. • Nginx Deployment can be exposed as a Load Balancer Service on Provider network. *(Optional, requires functioning Openstack Cloud-Provider.)* • Deploy a Statefull set using Openstack storage, provisioned through Kubernetes API. *(Optional, requires functioning Openstack Cloud-Provider.)*
Level	Summary
Trigger	On demand when requested by Team
Stakeholders	• DevOps Engineer • DevOps tool chain Tech Leads • Products teams Leads

Commented [VLD5]: Execute UC-0001 with input on number of master and worker nodes.
Add 'kubectl' (generic kubernetes CLI) and 'k3sup' (k3s configuration tool, https://github.com/alexellis/k3sup) to bastion host.
Install chosen Kubernetes distribution on Master nodes:
-Provision initial bootstrap node
-Provision and join secondary master node
-Provision and join third master node
Create worker nodes:
-For each worker node provision Kubernetes and join to cluster.
Verify internal access to k8s API:
-Export kube-config file to bastion host and verify functioning Kubernetes cluster using e.g. 'kubectl cluster-info' command.
Expose Kubernetes API:
-Create Openstack Loadbalancer to expose k8s API on Openstack Provider network
-Verify access to Kubernets API from the 'general DSV network'
Configure Kubernetes to use the Openstack Cloud-Provider (https://github.com/kubernetes/cloud-provider-openstack) enabling Kubernets to:
•Provision Openstack Loadbalancer (Octavia)
•Provision block-storage (Cinder)

Table 3: UC-0003 Build K8S cluster to run Private Cloud app