K8s Hands-on Labs

Before you begin

Please note that these materials are not a course, but rather set of scenarios which might test or expand your knowledge about K8s.

Knowledge on level of Linux Foundation Course is nice to have to complete all the tasks.

On the other hand materials are sorted in a way which allow to deep-dive into K8s world with basic knowledge and help of external resources like https://kubernetes.io/docs/home/

Scenario 1A) Prepare VMs

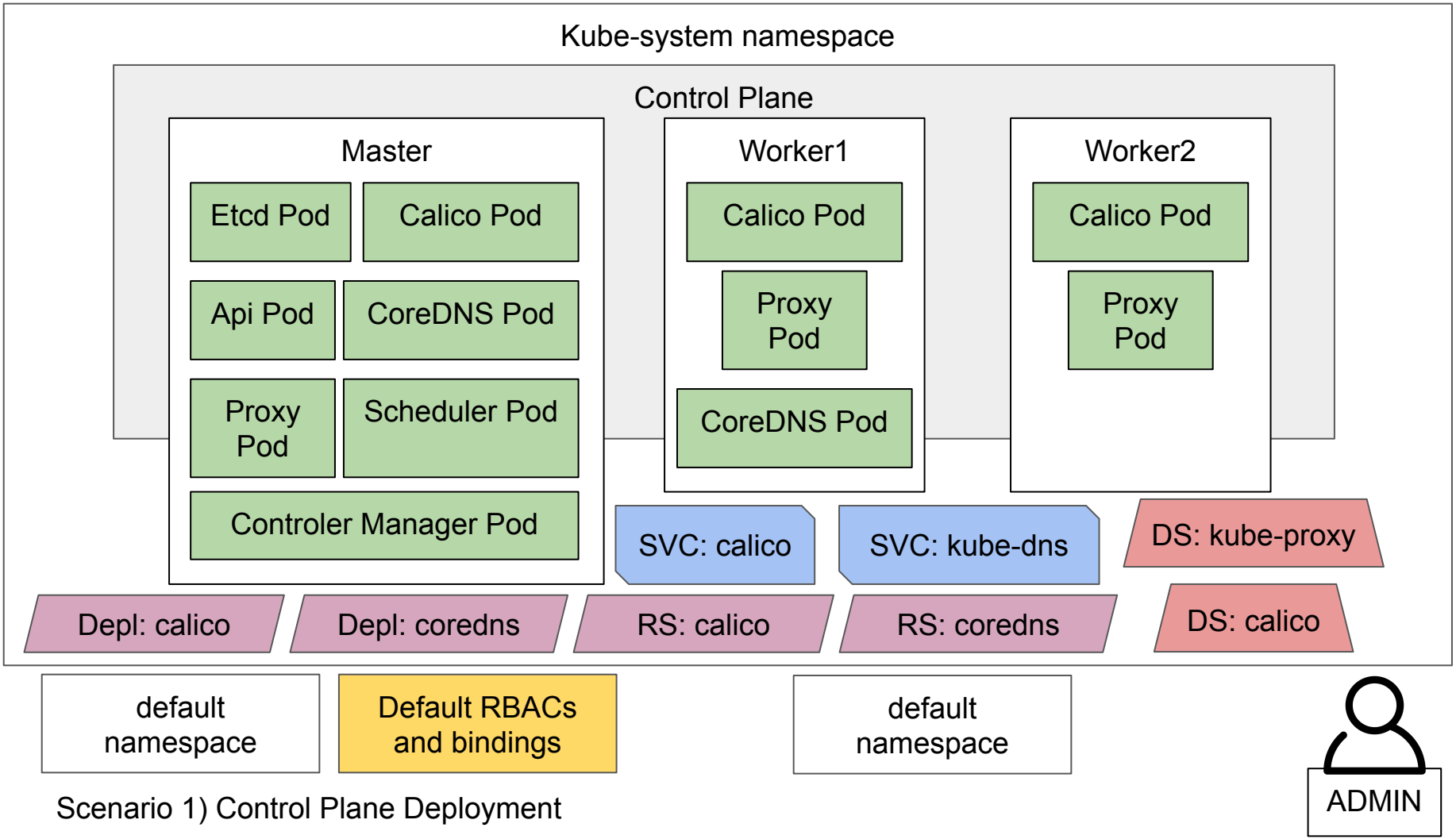
- Prepare lab environment using Vagrant and Vagrantfile available here: https://raw.githubusercontent.com/wiewioras/k8s-labs/master/Vagrantfile
- Vagrant will provide following VMs with 2Gigs of memory and 2vCPUs
- Proper internal networks will also be created and VMs will be accessible via following IPs:
  - 10.5.7.10 - Master Node with SSH keys for access to workers
  - 10.5.7.11 & .12 - worker1 and worker2
  - root user with password r00tme can be used to access the nodes
- All nodes have /etc/hosts file properly populated, and all needed packages installed and bash completion enabled for kubectl
- Following repo will be downloaded: https://github.com/wiewioras/k8s-labs on master node
- For more details see Vagrantfile

Scenario 1B) Deploy kubernetes cluster

- To deploy cluster use /root/init.sh script.
- Why you need to use “apiserver-advertise-address” with kubeadm init command?
- Why there is calico.patch?
- Use “kubeadm join” command returned by kubeadm on worker nodes to configure workers
- Make sure that cluster is working properly (kubectl get nodes)

Scenario 1 Questions:

- Which packages are needed to deploy Kubernetes
- What is a role of each control plane component
- How does kubeadm init works
- How to use TLS Bootstrap to add worker node
- What are manifests
- What are DaemonSets and how can this be used to expand control plane of K8s cluster



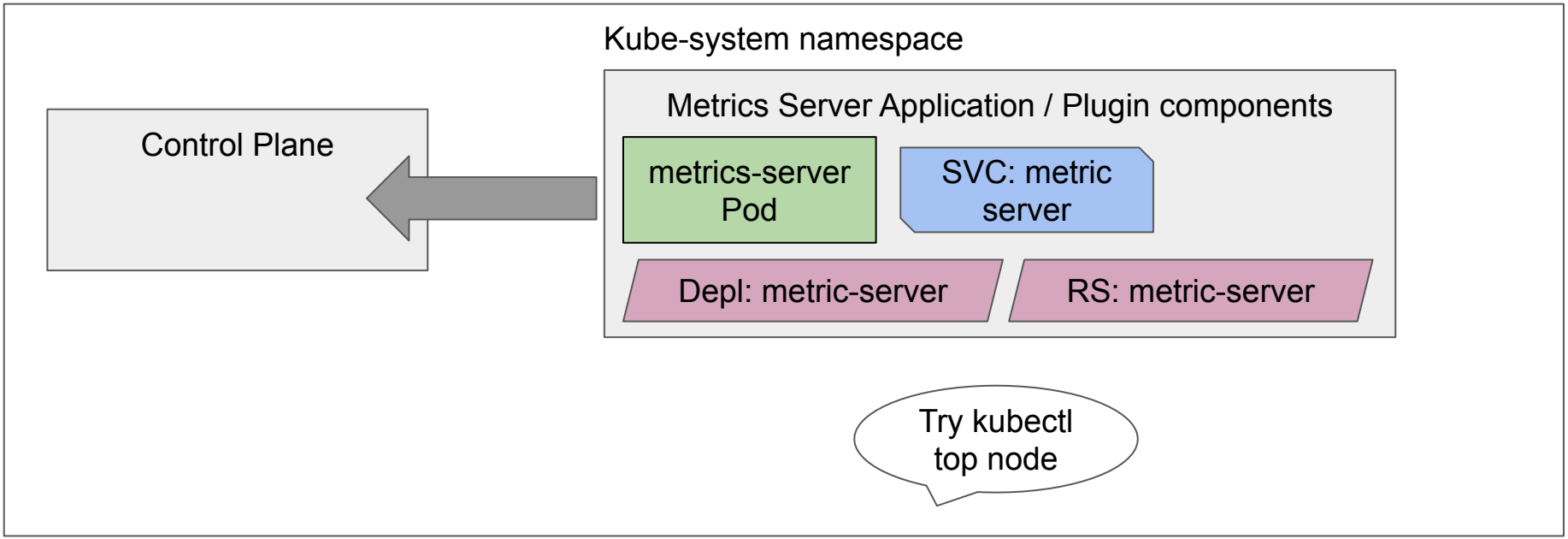
Scenario 1) Control Plane Deployment

Scenario 2) Deploy Metric server

- Go to /root/k8s/scenarios and familiarize with 2-* files
- Use 2-metrics-server.sh to deploy metrics server
- Why is metrics.patch used

Scenario 2) Questions

- How is metrics server used
- How are metrics collected
- What is kube-system namespace and in general describe kubernetes namespaces concept
- What objects were created to provide metrics server functionalities



default namespace

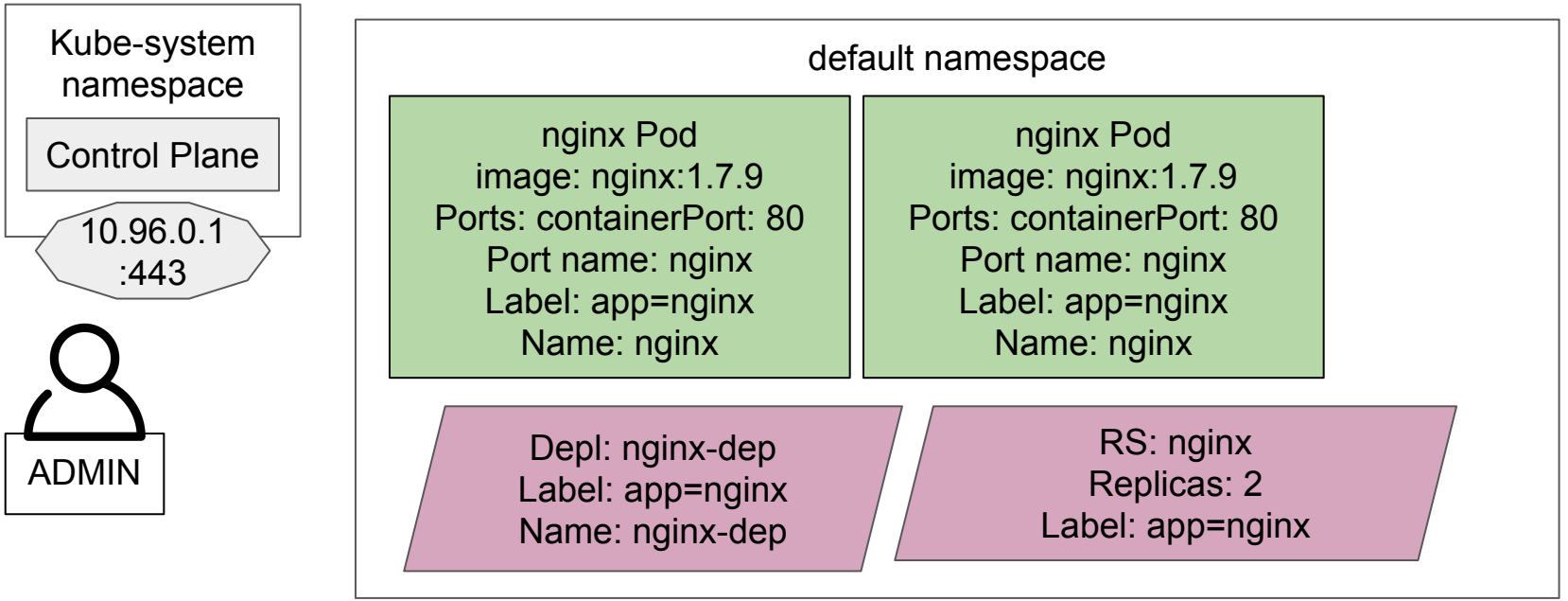
Scenario 2) Deploy Metric server on Master Node (Taints)
Note: From now on all control plane objects will be simplified into one object callect Control Plane.

Scenario 3) Basic deployment

- Use 3-sum.yaml file to deploy some basic application

Scenario 3) Questions

- Understand concept of pods, replica-sets and deployments
- Role of controller-manager and kubelet in scheduling, deployment and lifecycle of pods, replica-sets and deployments
- How to use labels to identify related objects
- How to use deployments, deployment specific parameters
- How to access container and container logs
- How is internal communication between pods and containers provided
- Check if there are any network limits:
    - Access every container and try ping to other containers, including one from other namespaces (in e.g. kube-system)



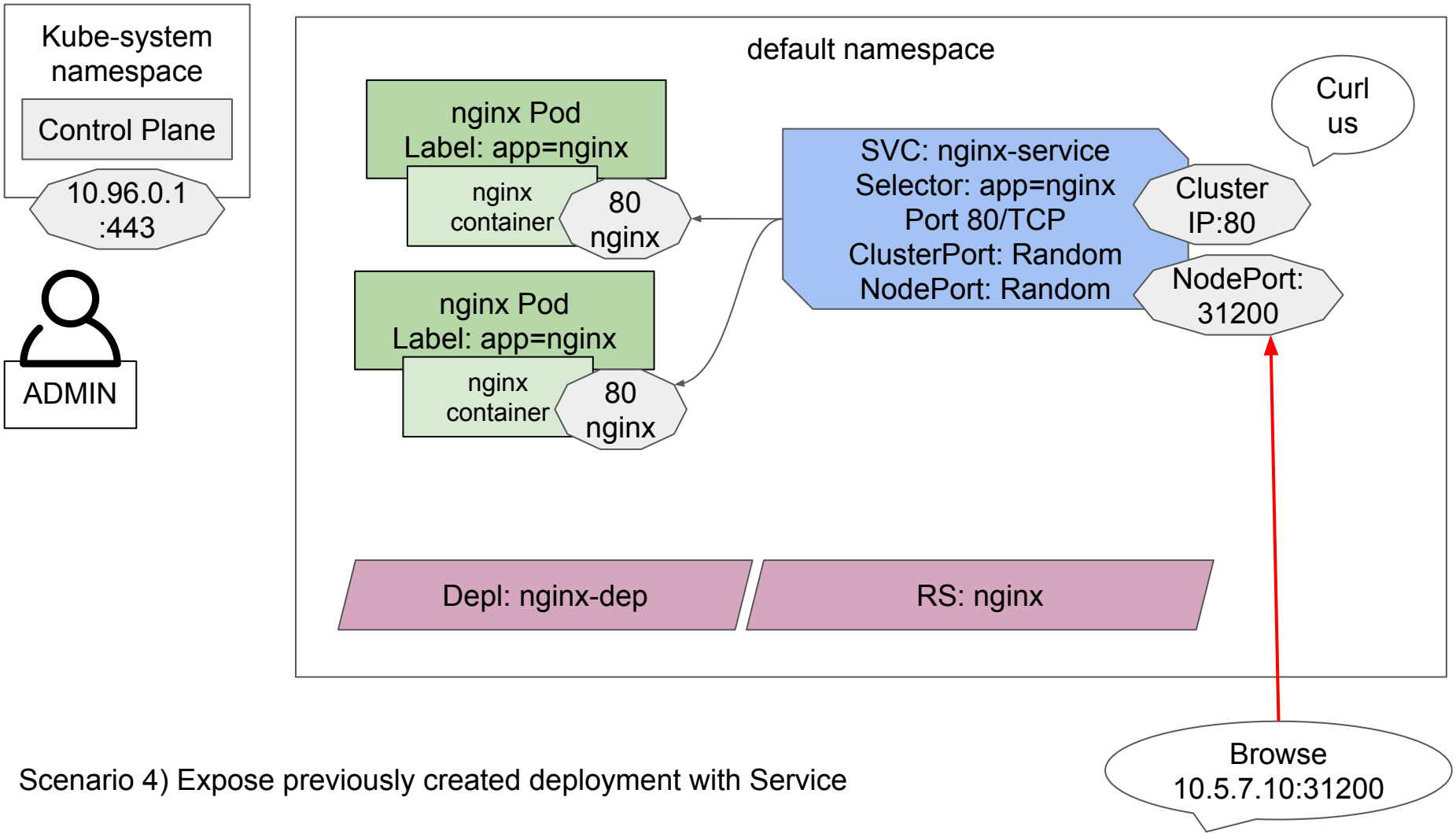
Scenario 3) Create simple deployment.

Scenario 4) Create service to access previously deployed application

- Use 4-sum.yaml file to deploy service which will serve nginx application
- What are possible parameters to be used to define/configure service

Scenario 4) Questions

- What is service
- What it a role of kube-proxy in serving service endpoints
- What is a difference between NodePort and ClusterIP



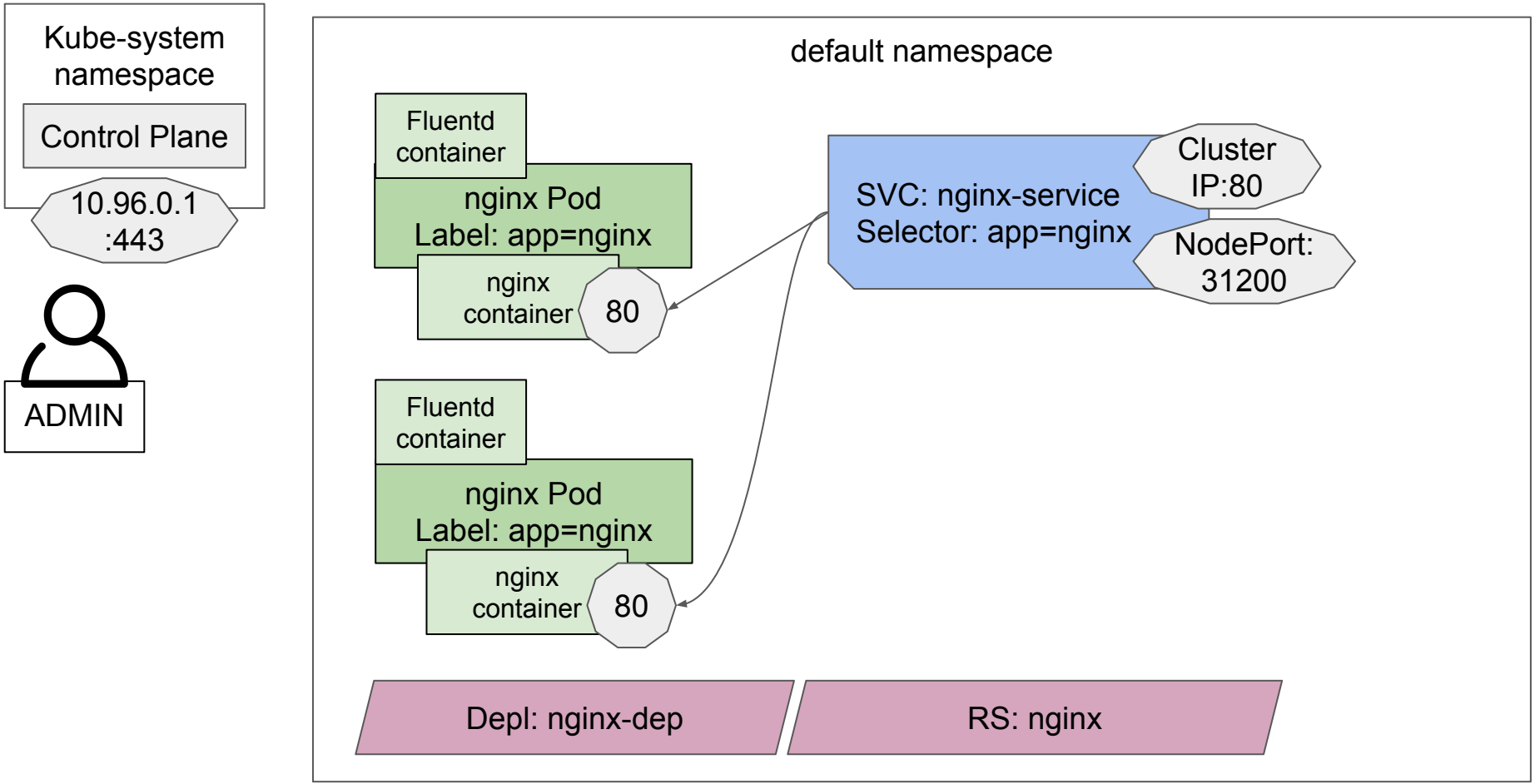
Scenario 4) Expose previously created deployment with Service

Scenario 5) Create multicontainer deployment

- Use 5-sum.yaml file to deploy extend previous application with additional fluentd container
- Collect information about possible usage of multi-container pods (ambasador, adapter, sidecar)

Scenario 5) Questions

- How to define multi-container pods
- How multi-container pods are scheduled, where containers are deployed
- How to access specific containers in multi-container pods (exec and logs)



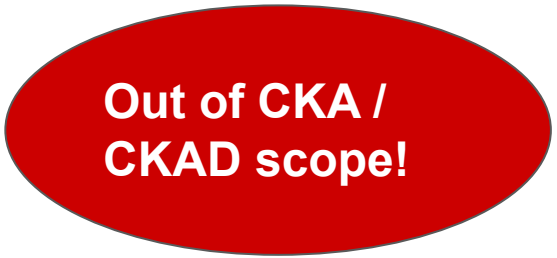
Scenario 5) Modify previous deployment: add fluentd containers to the pod (no specific configuration - just simple container deployment).

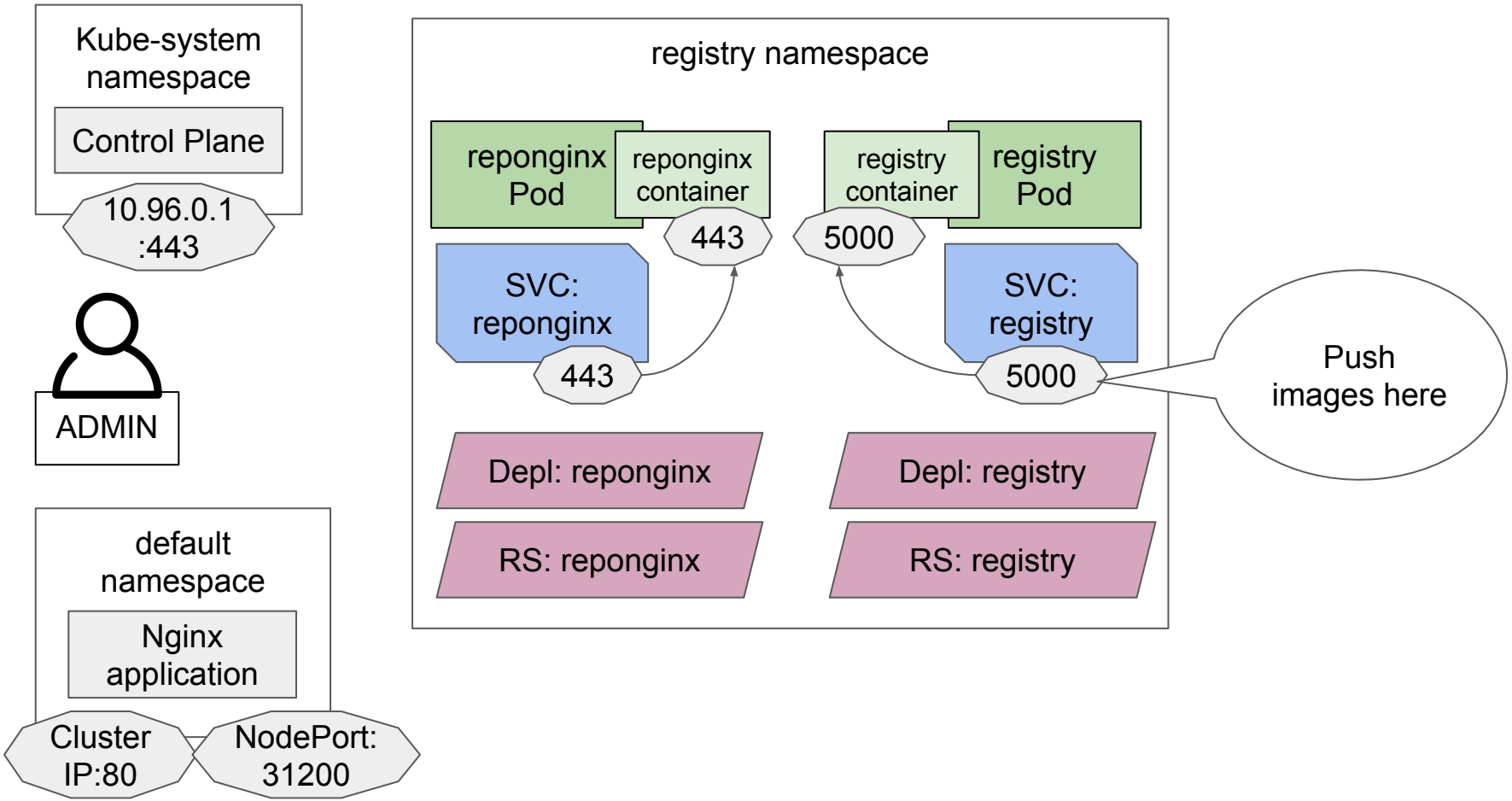
Scenario 6) Docker registry and images, basic DNS resolution

- Deploy local repo - use “6-deploy-registry.yaml”
- Get info about exposed repo service and check if it returns and empty list after
  $ curl http://<cluster_ip>:5000/v2/
- Add registry.registry.svc.cluster.local and its IP to /etc/hosts of every node (More details in 6-docker-images file)
- Allow docker to use your repo (More details in 6-docker-images file)
- Build alpine bastion image which will be used in later scenario (scenario 13)
- Tag image and push it to your local private repo

Scenario 6) Questions

- Familiarize with kubernetes deployment of local docker repository
- Familiarize with Dockerfile structure and how to build images
- Familiarize with tagging and naming concept for images
- Is it possible to build image without registry? How can you propagate it to other nodes?





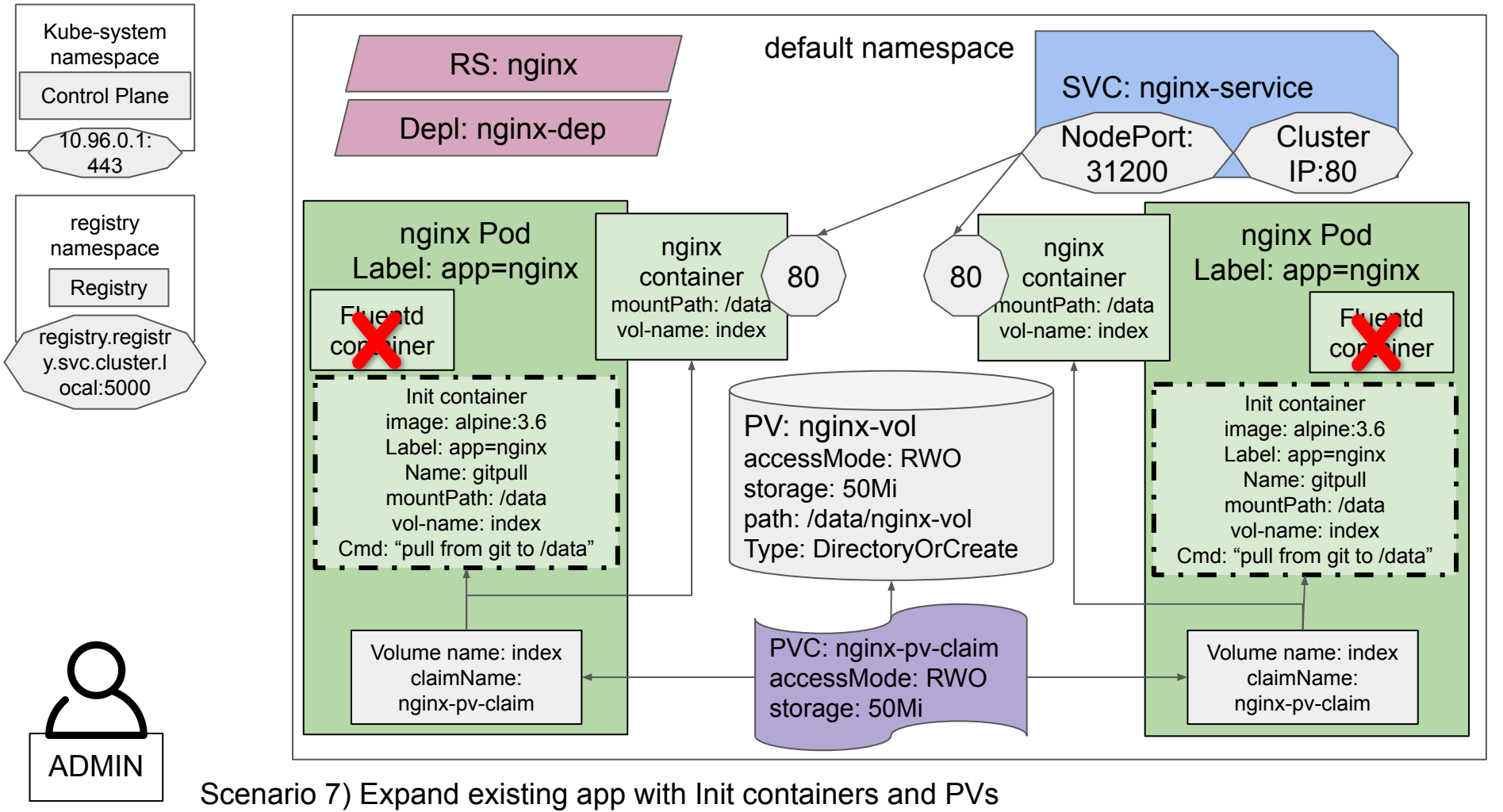
Scenario 6) Deploy Docker Repository

Scenario 7) Init containers, volumes and persistent storage

- use “7-sum.yaml” to improve previous app with persistent storage and init containers
- Make sure that you understand the concept of volumes (non-persistent), persistent volumes, and persistent volume claims

Scenario 7) Questions

- How can init containers be used
- How to pass different commands / entrypoints to containers
- What is storage class?
- What are other possibilities for providing persistent storage (ceph? ECS?)
- Do you know how to mount local data to specific container via persistent volume, proper claim, and storage class?



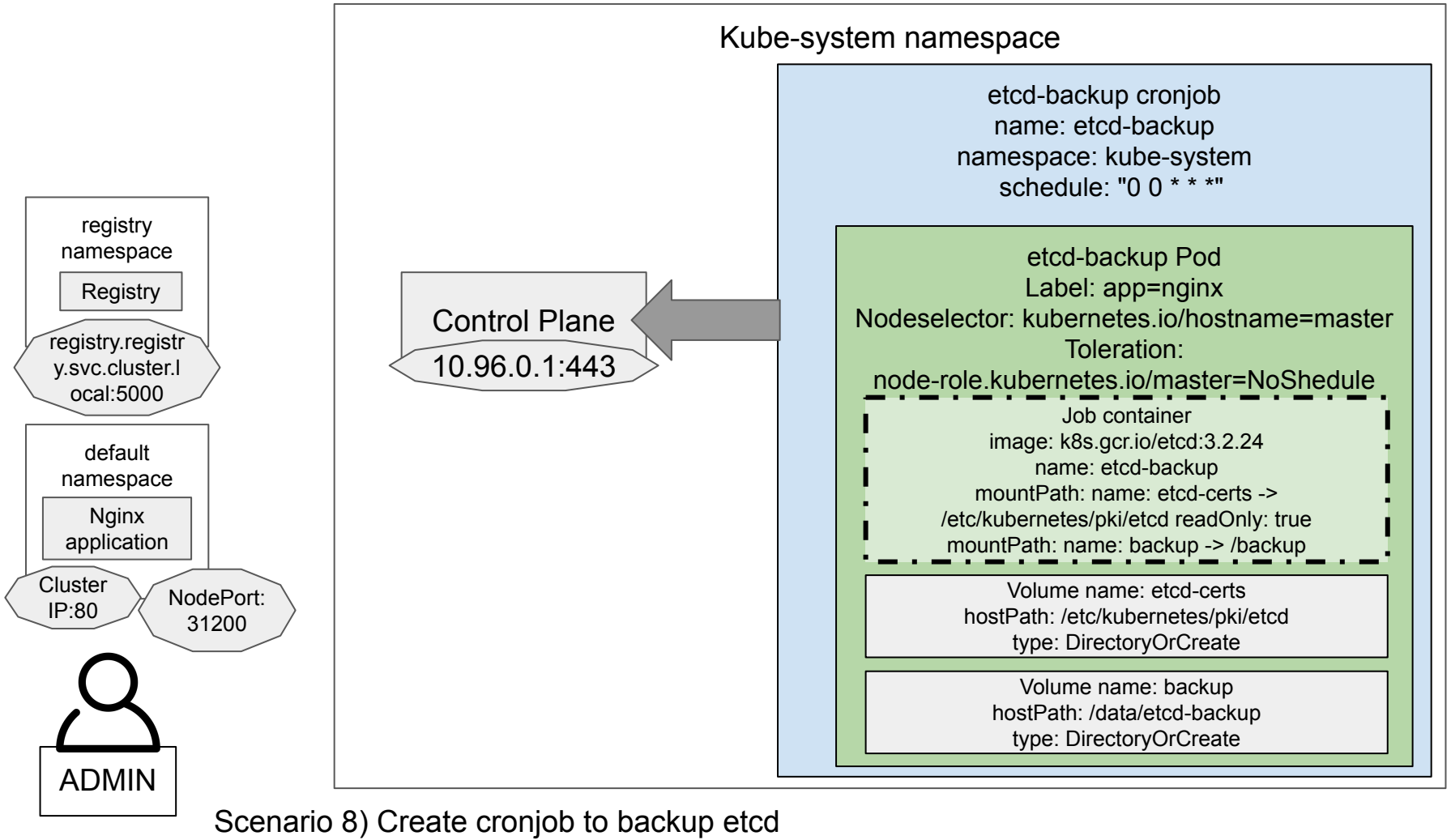
Scenario 7) Expand existing app with Init containers and PVs

Scenario 8) Jobs and CronJobs in Kubernetes

- Since this scenario we will leave nginx application we build previously untouched.
- This scenario covers simple usage of cronjob to perform containerized task of ETCD backup
- Use “8-etcd-backup.yaml” to deploy cronjob.
- By default backup will run at midnight. Modify cronjob parameters to run it sooner, and check how it works.
- Verify that backup was created in /data/etcd-backup localpath of master node

Scenario 8) Questions

- Do you know how to use jobs and cronjobs?
- What other purposes you see for jobs and cronjobs?
- Do you understand how the backup of ETCD is performed?



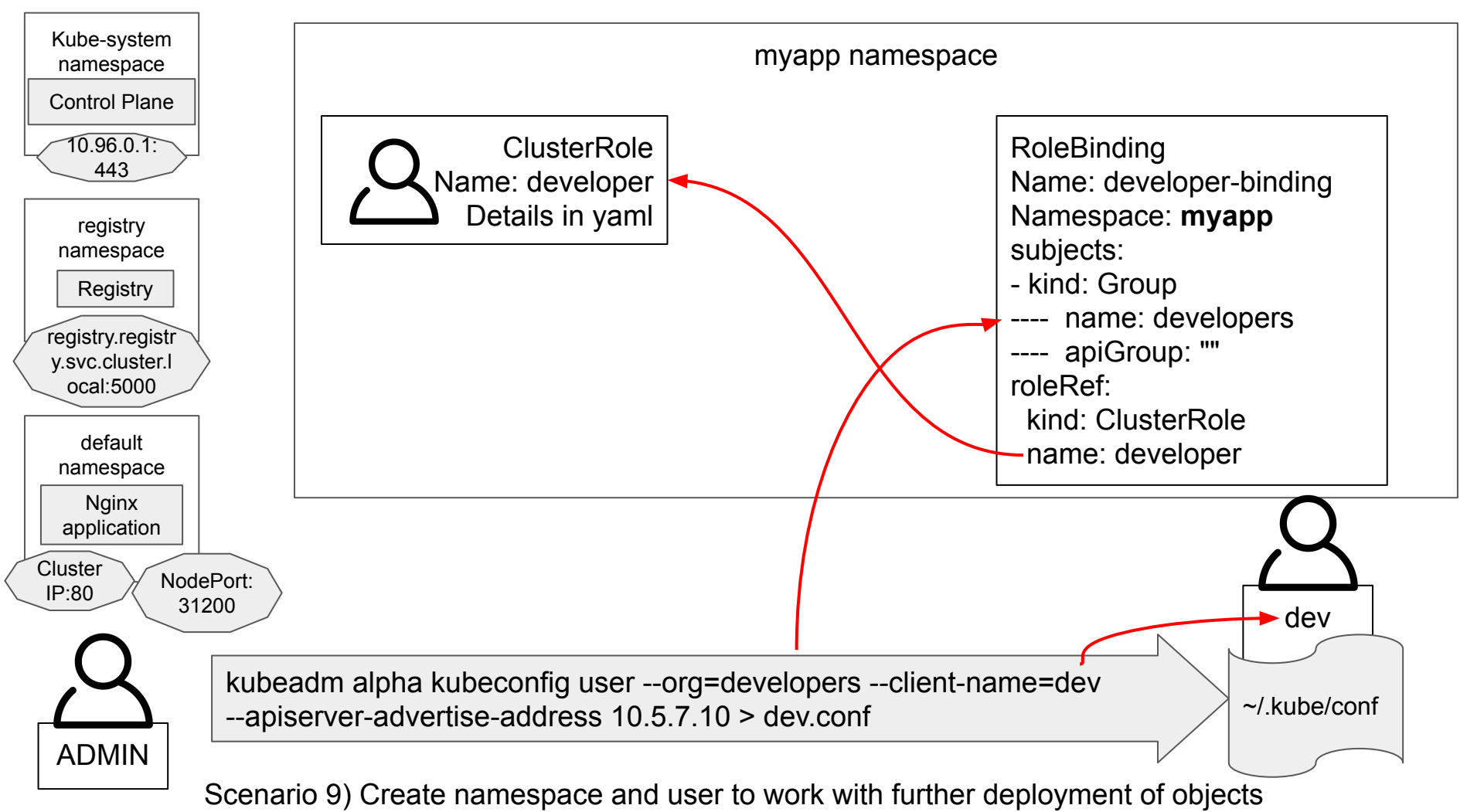
Scenario 8) Create cronjob to backup etcd

Scenario 9) Users management, RBAC

- Use 9-sum.sh bash script which will:
  - apply 9-users.yaml
    - To generate myapp namespace, ClusterRole named developer and RoleBinding of group developers to ClusterRole developer within namespace myapp
  - Create local system user and proper certificates using ‘kubeadm alpha kubeconfig user’. This user is only available on master node, and has password of r00tme. You can ssh to master node also as this user.
- From now on all task should be performed as dev user, and only specific actions should be performed by admin

Scenario 9) Questions

- Do you know how user is authenticated?
- What is kubernetes context?
- How to perform binding of user or group to specific roles
- What are differences between types of binding:
  - ClusterRole -> ClusterRoleBinding
  - ClusterRole -> RoleBinding (namespace)
  - Role -> RoleBinding
- What is the structure of certificate for user and service users?
- Are you familiar with this doc? https://kubernetes.io/docs/setup/certificates/



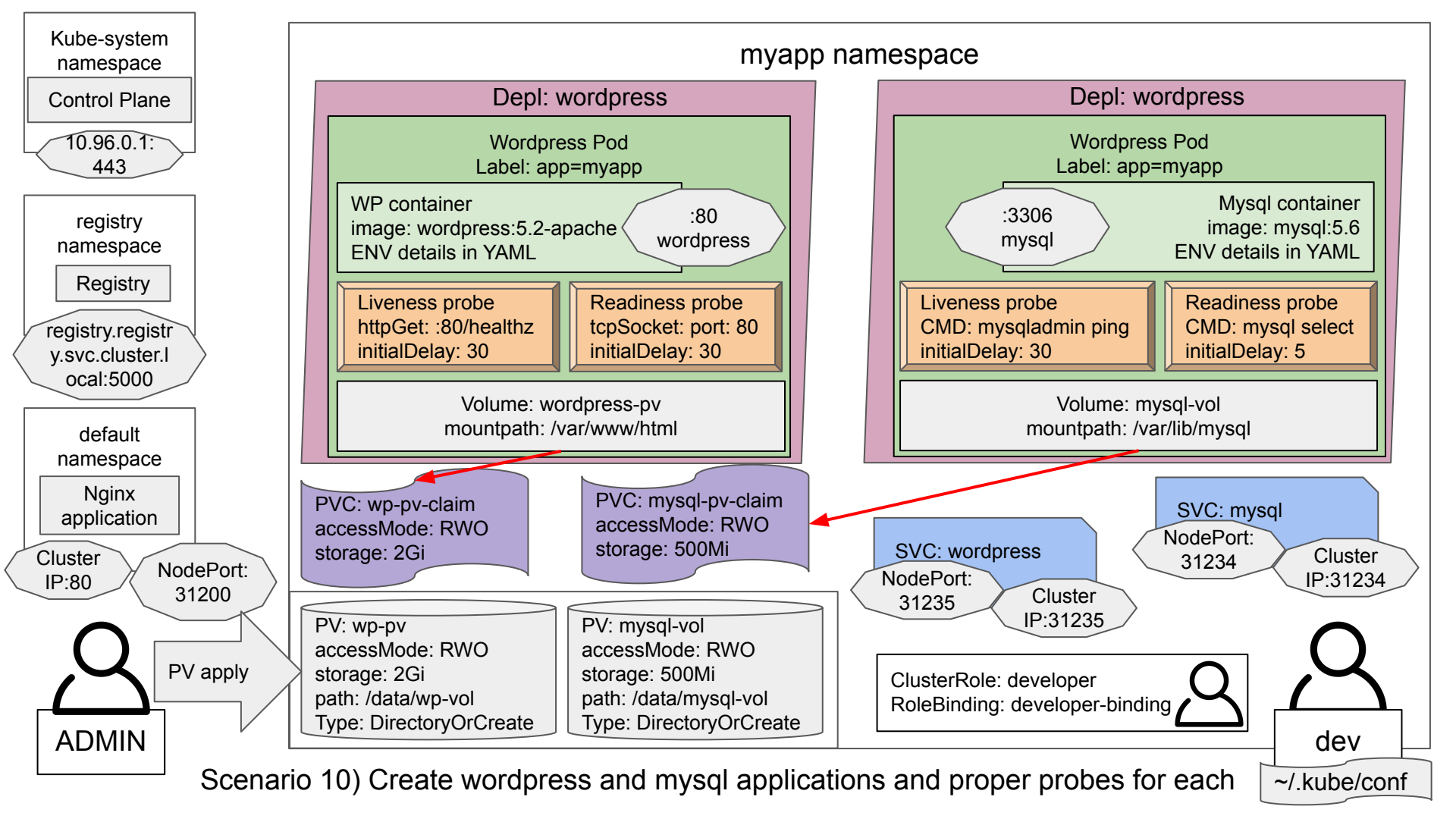
Scenario 9) Create namespace and user to work with further deployment of objects

Scenario 10) Probes

- As Admin use 10-sum-admin.yaml to create persistent volumes
- As dev use 10-sum-user.yaml to deploy new WordPress application and MySQL which is briefly described on next slide
- Probes will make sure that containers are ready (properly rolled out), and that are functioning without issues (liveness probes).

Scenario 10) Questions

- Are you familiar with content of 10-sum-user.yaml file?
- Do you know how to configure different types of probes?



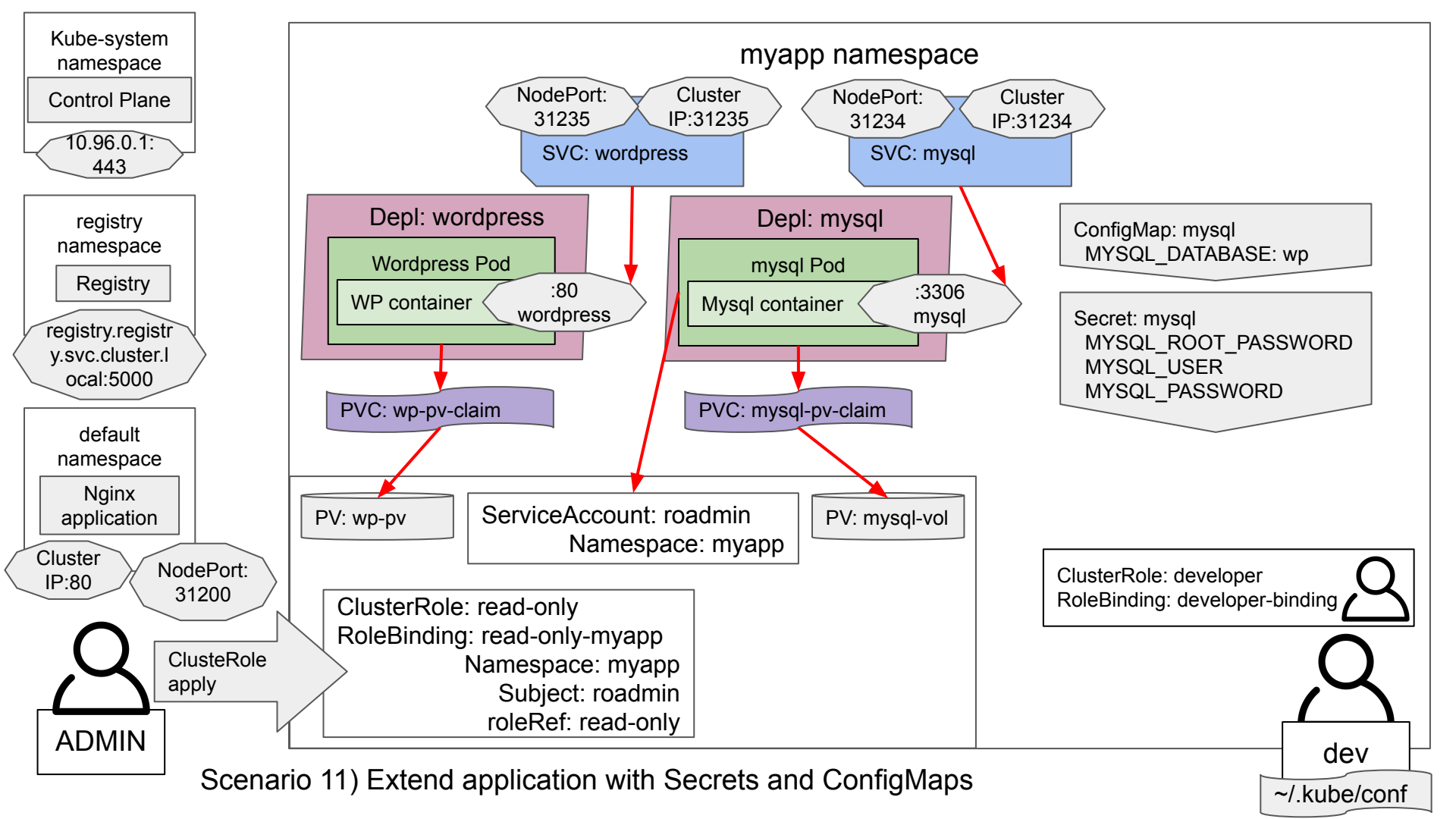
Scenario 10) Create wordpress and mysql applications and proper probes for each

Scenario 11) ConfigMaps. And Secrets

- As Admin use 11-sum-admin.yaml to create persistent volumes
- As dev use 11-sum-user.yaml to extend WordPress application and MySQL with ConfigMaps and Secrets
- Configmaps and Secrets will be used to pass environmental variables to proper containers

Scenario 11) Questions

- Are you familiar with content of 11-sum-user.yaml file?
- Do you understand how to create secrets (with base64 encoded data, and without it?)
- Make sure that you know how to pass data as environmental variables, and as files via volume mounting



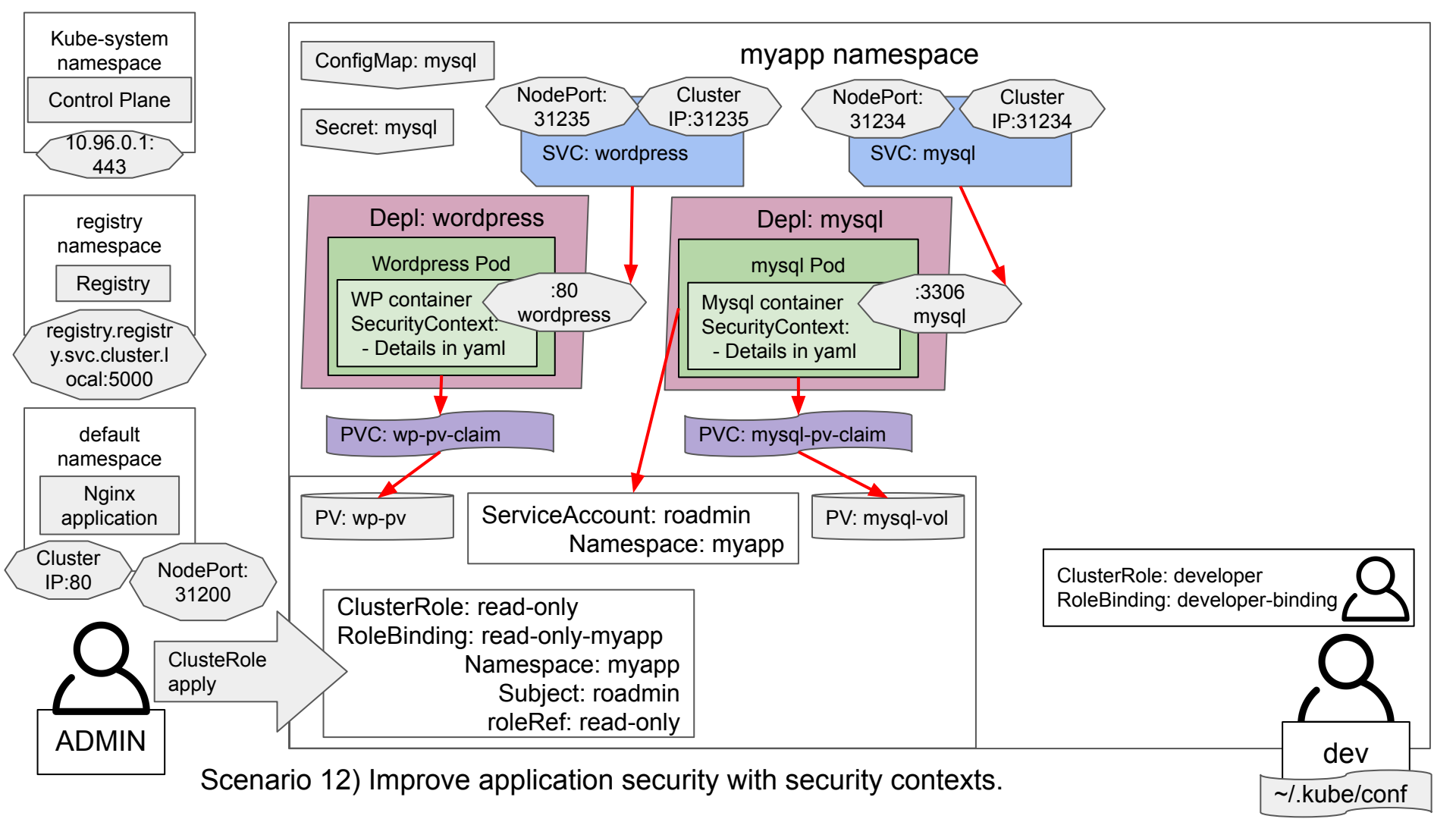
Scenario 11) Extend application with Secrets and ConfigMaps

Scenario 12) Security Context

- As dev use 12-sum-user.yaml to extend ordPress application and MySQL with Security Context
- In this task we will drop all linux capabilities and add only subset of these

Scenario 12) Questions

- What is default approach of K8s and Docker to users and Capabilities
- How to manage Linux Capabilities per pod
- How to run Pod as different user or as non-root? Will it work for every image?
- Can you give example of some of Linux Capabilities? (http://man7.org/linux/man-pages/man7/capabilities.7.html)



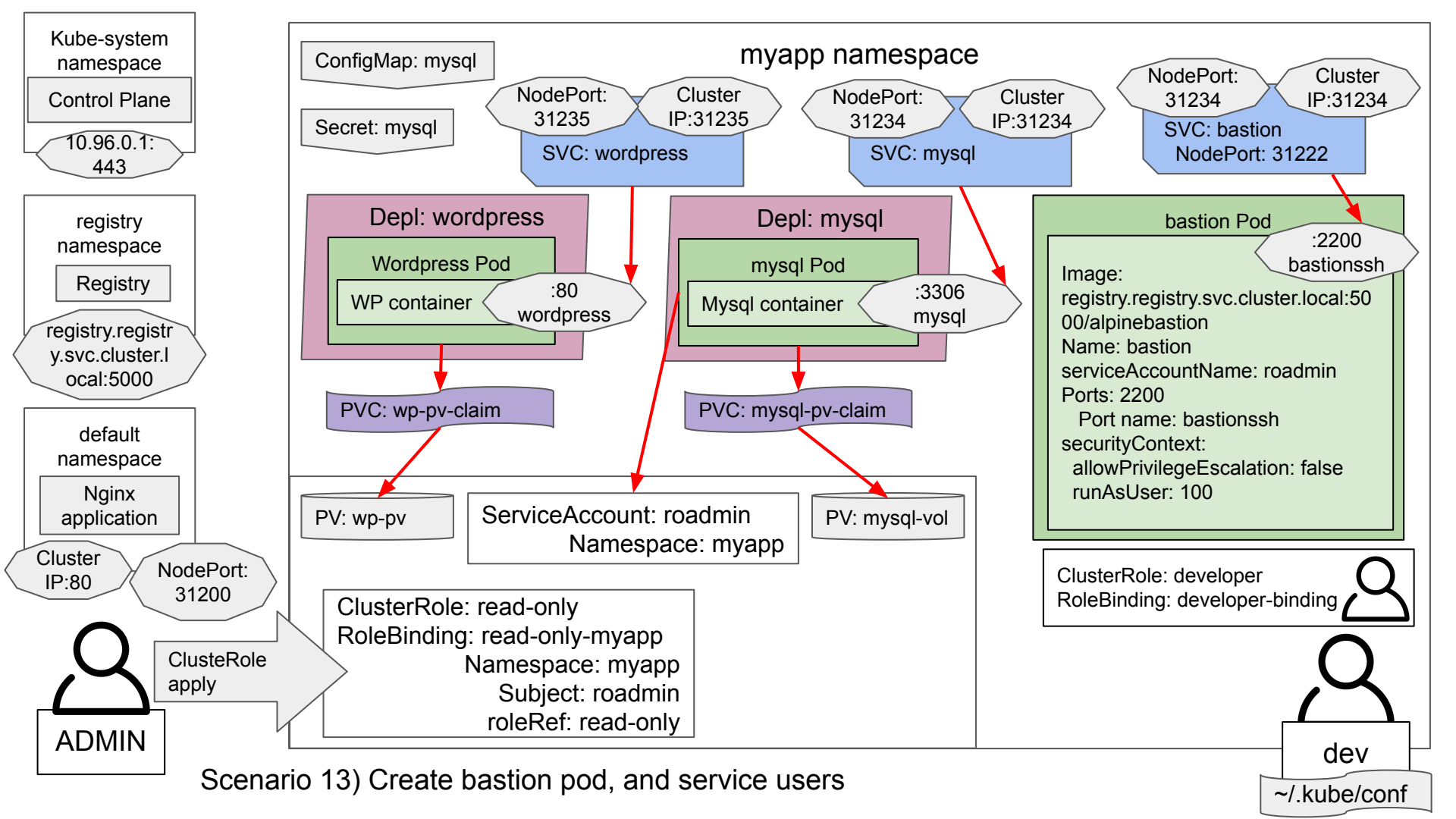
Scenario 12) Improve application security with security contexts.

Scenario 13) Service Users

- As Admin use 13-sum-admin.yaml to deploy roadmin Service Account, ClusterRole of read-only, and proper Role Binding for roadmin user to read-only role
- As dev use 13-sum-user.yaml to deploy Bastion Pod. Make sure that alpinebastion image from scenario 6 was built properly and is available in private repo.
- Bastion pod is nice example related to previous scenario where we are rebuilding alpine image, to have some basic workspace tools. It is also secured with dedicated non-root user, minimum subset of Linux Capabilities and can be run without privilege escalation.
- Bastion can be reached via ssh to exposed service. User is dev and private key can be found in scenarios/alpinebastion directory. Bastion is also deployed with proper contexts and certs installed, so user dev from bastion can authenticate against kubernetes API and be bind to developer group (in the same way as dev user from scenario 9)
- Bastion is also allowed to use roadmin kubernetes service user - token is mounted in /run/secrets/kubernetes.io/serviceaccount/token

Scenario 13) Questions

- Try using Kubernetes API from level of dev user on bastion.
  KUBE_TOKEN=$(cat /run/secrets/kubernetes.io/serviceaccount/token)
  curl -sSk -H "Authorization: Bearer $KUBE_TOKEN" \
  https://kubernetes.default:443/api/v1/namespaces/myapp/pods/$HOSTNAME
- Familiarize with Dockerfile and definition of Bastion pod - it briefly shows how kubernetes and docker images can (should?) be secured.



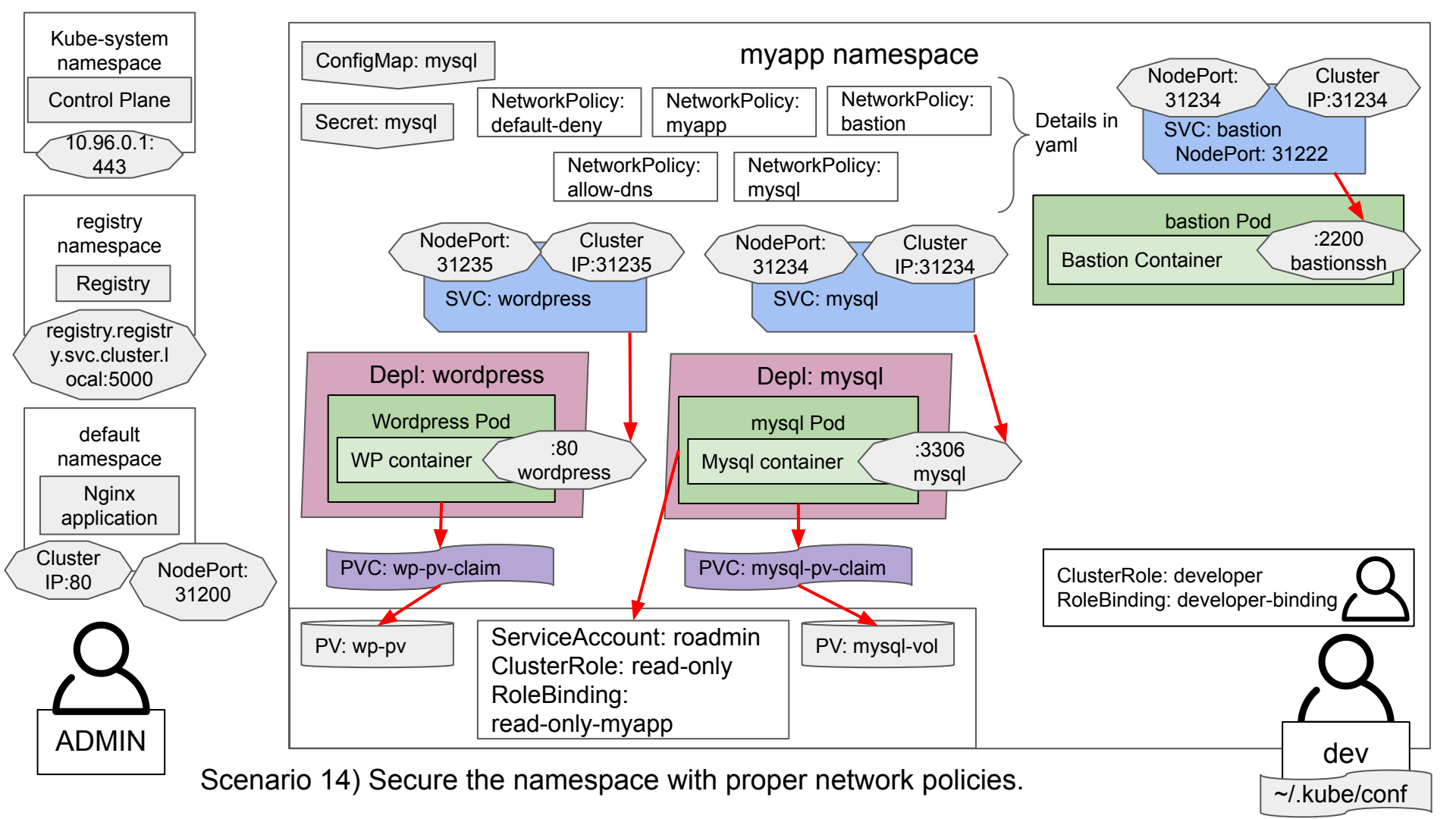
Scenario 13) Create bastion pod, and service users

Scenario 14) Network policies

- As dev user apply 14-sum-user.yaml to create proper network policies
- From now on bastion will be the only pod allowed to have full egress communication, and WordPress pod will be allowed only to reach MySQL server.
- Bastion will allow only SSH ingress, MySQL will accept ingress only on port 3306 (MySQL), and WordPress will allow ingress on 80
- All pods will be allowed to 53 UDP port (DNS)

Scenario 14) Questions

- Make sure that you understand how to create ingress and egress policies
- Do you know how to map specific policies to pods and namespaces?
- What is the scope of policies limitations (IP ranges, ports, specific IPs, hosts, etc.)



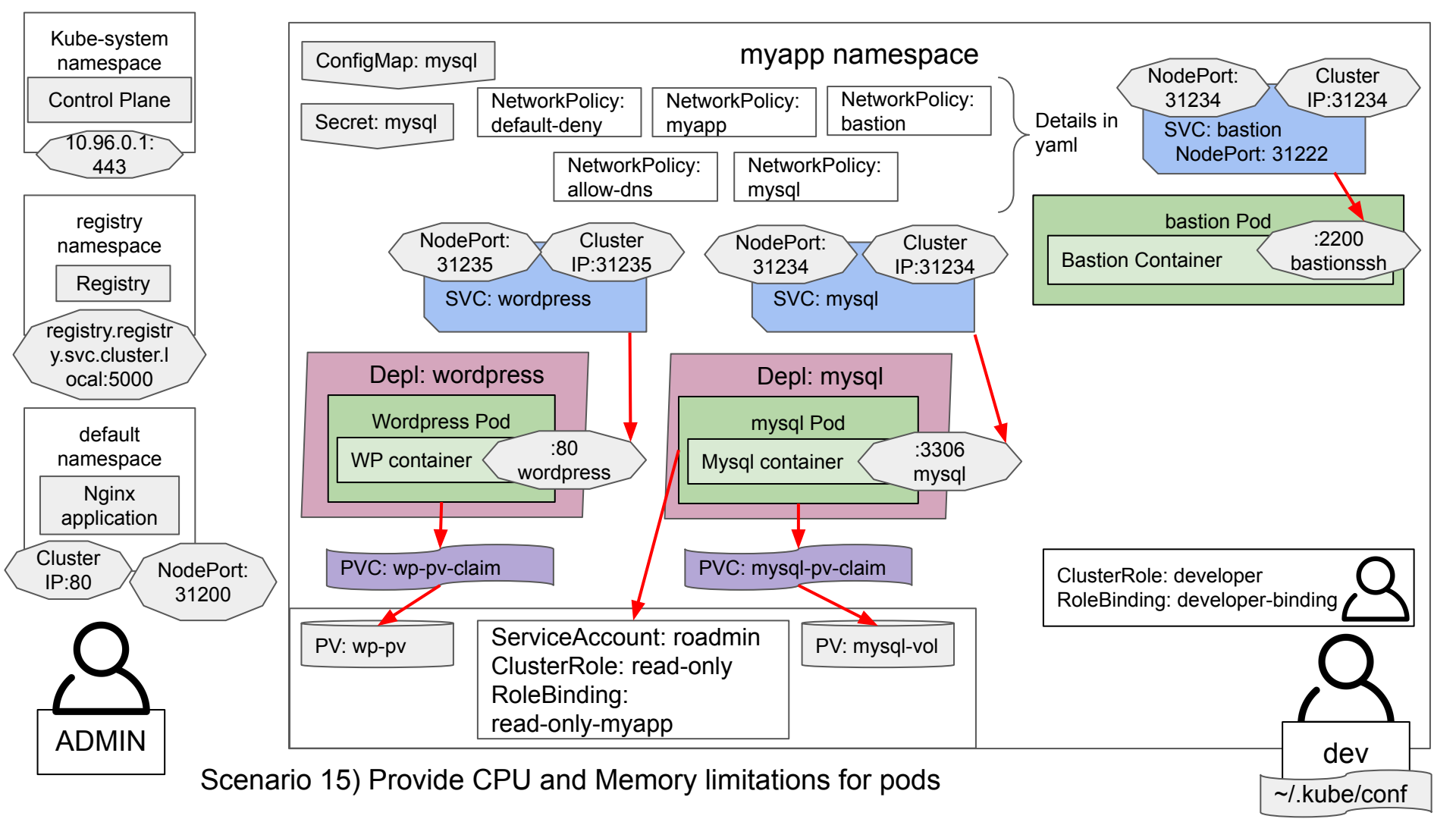
Scenario 14) Secure the namespace with proper network policies.

Scenario 15) Requests and Limits

- As dev apply 15-sum-user.yaml
- Above file will add memory and cpu limits and requests for all pods.

Scenario 15) Questions

- Make sure that you understand difference between limits and requests, how to use them, and how the scheduling is impacted by requests.
- Test limits putting some stress, e.g. via dd



Scenario 15) Provide CPU and Memory limitations for pods

What should I know?

Except of what was stated on previous slides, you should know how to:

- Get data about existing objects (get, describe, sorting, getting specific information via jsonpath, parameters like “show-labels” or “-o yaml”)
- Export existing objects into yaml (get -o yaml --export)
- Collect as much information from kubernetes.io site (use templates and examples available there).
- How does the basic elements of Control Plane works (DNS, scheduler, metrics, etcd)
- How kubernetes are deployed (basic configuration, certs needed for communication, systemd files, etc.)
- How to perform rolling updates, edit existing objects, analyse deployment history
- How to analyze logs of pods and understand statuses of K8s objects.

What to do next

- Ingress controller was not covered, although you can find some useful files in additional_materials directory
- Play with your existing application. Use bastion host wherever you can to simulate more production-like envrionment. Try not to use admin account, and create proper roles for actions you want to take.
- Try extending existing application with e.g. stateful set of MySQL galera in HA, deploy Ingress controller and define Ingress to access WordPress. Use ‘kubectl api-resources’ command to see if there are more K8s objects which can be used to enrich your application.
- Try deploying K8s in HA