Certified Kubernetes Administrator Course

Introduction (S1)

- Info’s about this course, the certificate and other courses
- Certificate:
    - 300$ + free retry
    - Online examen
    - Mostly practice tasks

Core Concepts (S2)

Cluster Architecture

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

ETCD for beginners

- Simple key value store
- Information stored as documents/pages for each entry (YAML, JSON, …)
- *./etcdctl set <key> <value>*
- *./etcdctl get <key>*

ETCD in Kubernetes

- Stores information about Nodes, PODs, Configs, Secrets, …
- Kubeadm does the installation for you and creates an etcd POD
- Multiple etcds should be connected to each other

Kube-API-Server

- Primary management component
- Kubectl commands work over the api server using POST and GET requests
- Responsible for:
  - Authenticate user
  - Validate request
  - Retrieve data
  - Update etcd
  - Scheduler
  - Kubelet
- Kubeadm deploys it as POD
  - Options under */etc/kubernetes/manifests/kube-apiserver.yaml*
- *Non kubeadm setup:*
  - */etc/sysdemd/system/kube-apiserver.service*
  - *ps –aux | grep kube-apiserver*

Kube Controller Manager

- Manages controllers in kubernetes
- Process that continuously monitors state of components of system and keeps system in desired state
- A lot of different types of controllers
  - All get installed with kubernetes Controller Manager
- Kubeadm deploys it as POD
  - Options under */etc/kubernetes/manifests/kube-controller-manager.yaml*
- *Non kubeadm setup:*
  - */etc/sysdemd/system/kube-controller-manager.service*
  - *ps –aux | grep kube-controller-manager*

Kube Scheduler

- Deciding which POD goes on which node and selects best Node
  - Depending on resource requirements, etc.
  - Can be customized
- Kubeadm deploys it as POD
  - Options under */etc/kubernetes/manifests/kube-scheduler.yaml*
- *Non kubeadm setup:*
  - */etc/sysdemd/system/kube-scheduler.service*
  - *ps –aux | grep kube-scheduler*

Kubelet

- create, delete containers on nodes with help of container runtime
- send information to master
- kubeadm does not install it automatically
  - *ps –aux | grep kubelet*

Kube Proxy

- service is virtual component only in kubernetes memory
- kubeproxy runs on each node and looks for new services and creates appropriate rules for network forwarding
- Kubeadm deploys it as POD on each node
  - Options under */etc/kubernetes/manifests/kube-proxy.yaml*
- *Non kubeadm setup:*
  - */etc/sysdemd/system/kube-proxy.service*
  - *ps –aux | grep kube-proxy*
-

POD's

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

PODS with YAML

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

Demo POD's with YAML

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/tree/master/Section%206

Recap ReplicaSets

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

Deployments

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

Namespaces

- Every namespace has its own resources and policies
- To reach other namespace you have to append namespace to the pod name
- Kubectl get pods –namespace=<namespace>
    - Kubectl get pods –all-namespace
- Kubectl create -f <yaml-file> –namespace=<namespace>
    - Or add namespace to metadata
- Create namespace with definition file and kubectl create
    - Or kubectl create namespace <namespace>
- Change default namespace:
    - Kubectl config set-context $(kubectl config current-context) –namespace=<namespace>
- Resource quota to limit resources in namespace
    - With definition yaml file

Services

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

Services Cluster IP

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

Scheduling (S3)

Manual scheduling

- Scheduler searches for Pods without nodeName
    - Selects best node and sets the nodeName for the Pod
    - nodeName can be set manually in definition file
    - another way is with pod binding object (via definition yaml) and send POST request

Labels and Selectors

- labels: properties attached to items
    - added in definition file under metadata
- selectors: filters based on labels
    - *kubectl get <object-type> --selector <label>=<label-value>*
    - also used in replicasets, deployments, services, etc.
- annotations: additional information added to a object
    - added under metadata

Taints and Tolerations

- used to set restrictions to schedule which pods can go on which nodes
    - will not guarantee that a pod goes on a certain node
- Nodes have Taints
    - By default no taints on worker nodes
    - Master node has a default taint
    - *Kubectl taint nodes <node-name> key=<value>:<taint-effect>*
- Pods can have Tolerations
    - Default no tolerations
    - Specified in definition file under spec→tolerations

Node Selectors

- Limitation for pods to specify nodes
    - In pod definition file under spec > nodeSelector
    - Works with labels on nodes

Node Affinity

- To ensure pods are hosted on particular nodes
    - Specified under spec > affinity > nodeAffinity in pod definition
    - More complex than node selectors, also more options
- requiredDuringSchedulingIgnoredDuringExecution
    - node has to match or pod will not be scheduled
    - affinity ignored during execution
- preferredDuringSchedulingIgnoredDuringExecution
    - matching node is preferred but other nodes are used either
    - affinity ignored during execution
- requiredDuringSchedulingRequiredDuringExecution
    - node has to match or pod will not be scheduled
    - node has to match or pod will not be scheduled during execution

Taints and Tolerations vs Node Affinity

- for some cases taints and node affinity has to be used in combination

Resource Requirements and Limits

- scheduler taks resources in consideration for deploying pods
    - if not enough resources in any node POD does'nt get deployed
- Default Requests: 0.5 CPU (1=1vCPU), 256 Mi Memory, disk
    - Can be specified in deployment definition under spec > resources > requests
- Default Limits: 1 CPU, 512 Mi
    - Can be specified in deployment definition under spec > resources > limits
    - Limits can't be exceeded for CPU

DeamonSets

- Like replica sets but runs one copy of a POD on each node of the cluster
- Good for monitoring, logging, kube-proxy, networking, etc.
- Similar to replicaset definition just kind is different
- Uses default scheduler and nodeAffinity

Static Pods

- Nodes can be managed independently with kubelet
    - Pod definitions have to be placed under /etc/kubernetes/manifests
        - Path is option of the service pod-manifest-path or in kubeconfig.yaml
        - Only pods can be created like this
    - Kubelet tries to keep pods alive
    - Pods can be viewd with docker ps

Multiple Schedulers

- You can deploy your own scheduler
    - Can be specified with definition yaml too
- Cluster can have multiple schedulers
    - You can specifiy the scheduler when deploying a Pod
        - In the definition under spec > schedulerName
- View events with kubectl get events
- Kubectl logs <scheduler-name> --name-space<namespace>

<u>Logging & Monitoring (S4)</u>

<u>Monitor Cluster Components</u>

- Kubernetes doesn’t have own monitoring solution
    - Other open source solutions like Prometheus, elastic stack, etc. work
        - Metrics Servers
        - As in memory solution
    - Kubelet cAdvisor retrieves performance metrics and exposes them to kubeApi
    - Get from github and deploy with kubectl create
    - *Kubectl top node* to view metrics of nodes
    - *Kubectl top pod*

<u>Managing Application Logs</u>

- Docker: docker logs [–f] <containerid>
- Kubernetes:
    - kubectl logs [–f] <pod-name>
    - with multiple containers: kubectl logs [–f] <pod-name> >containerid>

ALM (S5)

Rolling Updates and Rollbacks

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

Commands

- Add commands with cmd or with a entrypoint or combination of both
    - FROM Ubuntu Entrypoint [“sleep”] CMD [“5”]

Commands and Arguments

- Add arguments under pod-definition containers>container>args
- Add commands under pod-definition containers>container>command

Configure Enviroment Variables in Application

- Set them under containers>container>env
    - List with name value pairs

Configure ConfigMaps in Application

- Used to pass config data from a file for env variables
    - Create config map and then incject in pod
    - *Kubectl create [-f] configmap [definition file]*
    - *Kubectl get configmaps*
    - *Kubectl describe configmaps*
    - Add to pod with envFrom

Configure Secrets in Application

- Used to store sensitive information
- Similar to config maps but get encoded
    - Encode data with echo –n ‘<string to encode>’ | base4
        - Decode with echo –n <encoded-string> --decode | base64
    - *Kubectl create [-f] secret [definition file]*
    - *Kubectl get secrets*
    - *Kubectl get secret <secret> -o yaml*
    - *Kubectl describe secrets*
    - Add to POD with envFrom –secretRef or mount as Volume

Multi Container PODs

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf

<u>Cluster Maintenance (S6)</u>

<u>OS Upgrades</u>

- Pod eviction timeout… time until pod is considered dead if it not responds
    - *Kube-controller-mager –pod-eviction-timeout=5m0s*
- Drain nodes to get pods to other nodes *kubectl drain <node-name>*
    - *Kubectl uncordon <nodename>* to remove drain
    - *Kubectl cordon <node-name>* to make node unschedulable

<u>Kubernetes Software Versions</u>

- See version with *kubectl get nodes*
    - Major.minor.patch, –aplha and –beta releases also available

<u>Cluster Upgrade Process</u>

- Scheduler and controller-manager can be a version lower than apiserver
- Kubelet and kube-proxy can be 2 versions lower than apiserver
- Kubectl can be a version higher or a version lower than apiserver
- Kubernetes only supports latest 3 versions
- Update one minor version at a time
- Update easy on cloud platform
- Kubeadm Update:
    - *Kubeadm upgrade plan*
    - *Apt-get upgrade –y kubeadm=<version>*
    - *kubeadm upgrade apply <version>*
    - *Apt-get upgrade –y kubelet=<version>*
    - *Systemctl restart kubelet*
- First upgrade master than the workers
    - Strategy1: Upgrade all at ones
    - Stretegy2: Upgrade one node at a time
        - Kubectl drain <node>
        - *Apt-get upgrade –y kubeadm=<version>*
        - *Apt-get upgrade –y kubelet=<version>*
        - *Kubeadm upgrade node config –kubelet-version <version>*
        - *Systemctl restart kubelet*
        - *Kubectl uncordon <node>*
    - Strategy3: add new nodes with new version, move workload and remove old node

<u>Backup and Restore Methods</u>

- Resource Configs:
    - Declarative configuration preferred way and store them in a src-repository
    - Backup all: kubectl get all –all-namespaces –o yaml > all-deploy-services.yaml
- ETCD-Cluster:
    - Etcdctl snapshot save <name>
    - Etcdctl snapshot status <name>
    - Etcdctl snapshot restore <name> --data-dir <directory> afeter stopping kube-apiserver service, config service, reload deamon, restart etcd, restart kube-apiserver

Security (S7)

Kubernetes Security Primitives

- Hosts should be secured (no root access, pw based authentication disabled, only ssh)
- Kube-apiserver should be secured with authorization
  - Everything using tls encryption
- Restrict access between pods

Authentication

- Kubernetes does not handle users internally
  - Serviceaccounts can be managed with kubernetes
- Kube-apiserver auth mechanisms
  - Static password file
    - Eg. Csv. File with pw, username, userid, [group] and specify file in kubeapiserver service
    - Not recommended
  - Static token file
    - Instead pw use a token
    - Not recommended
  - Certificates
  - 3rd party identity services

TLS Basics

- Guarantee trust between parties in a transaction
- Asymmetric encryption(public key and private key) PKI(public key infrastructure)
  - Generate with ssh-keygen
  - On Server: openssl genrsa –out <key> <base>
  - Certificate to verify validity of key transfer (client and server certificates)
  - CA organizations sign certificates with their public and private keys (root certificates)
    - You can host private CA instances
  - Browsers check certificates
  - Server can request from client

TLS in Kubernetes

- Communication between nodes, components and users needs to be secured
- Servers:
  - Kube-apiserver exposes http service and requires server certificate
  - Etcd-server has requires own server certificate
  - Kubelet servers require server certificates too
- Clients:
  - Users like admin
  - Kube-scheduler
  - Kube-controller-manager
  - Kube-proxy
  - Kube-apiserver with etcd-server
  - Kube-apiserver with kubelet-server

TLS in Kubernetes – Certificate Creation

- Different tools like easyrsa, **openssl**, cfssl
- Kubeadm does most of the work for you
- CA Certificate:
    - Generate Key: *openssl genrsa –out <key-name> 2048*
    - Certificate Signing Request*: openssl req –new –key <keyname> -subj "/CN=KUBERNETES-CA" –out <keyfile>*
    - Sign Certificates: *openssl x509 –req –in <keyfile> -signkey <keyname> -out <certname>*
    - CA certificates files have to be well protected
- Client Certificates:
    - Admin user:
        - Like CA certificate with different names and "CN=kube-admin/O=system:masters"
        - Openssl x509 –req –in admin.csr –CA ca.crt –Cakey ca.key –out admin.crt
        - Specifiy certificate in kube-config.yaml
    - Similar process for other client certificates
- Server Certificates
    - Etcd-server:
        - Similar to other certificates
        - Needs additional peer certificates
    - Kube-apiserver:
        - Create openssl.cnf with dns and IP's and path as option while generating
    - Kubelet-server:
        - Named after nodes and use them in kubelet-config.yaml
        - E.g.: Name: system:node:node01
        - Needs to be done for each node

View Certificate Details

- See spreadsheat in directory for what to check
- Cat /etc/kubernetes/manifests/kube-apiserver.yaml
- *openssl x509 –in <cert file> -text –noout* to view details
- inspect service logs for problems if kubeapi or etcd is down use docker logs

Certificates API

- Certificate signing request is send trough certificates API to CA server
    - Admin can approve request and share cert with user
    - Controller Manager does this
    - *Openssl genra –out <key> 2048*
    - *Openssl req –new –key <key> -subj "/CN=<username>" –out <file>*
    - Request specified with yaml file
    - *Kubectl get csr <request> -o yaml*
    - *Kubectl certificate approve <request>*

<u>Kube Config</u>

- Use kubeConfig file to store cert and key configurations and use the file for api requests
- 3 Sections:
  - Clusters: clusters you have access too
  - Users: users accounts you have you
  - Contexts: specify which user you want to use for which cluster
    - Namespace can also specified here
- *Kubectl config view [-- <config file>]*
- *Kubectl config use-context <user@cluster>*

<u>API Groups</u>

- Core group (/api):
  - Namespaces, pods, events, nodes, bindings, …
- Named group (/apis):
  - /apps, /extensions, /networking.k8s.io, … with their resources with their actions
  - See them under *curl <server-adress>:6443-k*
    - Use certificate files to authenticate
    - Or use *kubectl proxy*

<u>Role Based Access Control</u>

- Specified in role definition yaml
  - Role binding yaml object needs to be specified too
- *Kubectl get/describe roles*
- *Kubectl get/ describe rolebindings*
- Check access with:
  - kubectl auth can-i <do something>
  - kubectl auth can-i <do something> as <user>

<u>Cluster Roles and Role Bindings</u>

- see namespaced resources: kubectl api-resources –namespaced=true
- cluster roles/bindings for cluster scoped resources
- similar to non-cluster based roles
- can be used for namespace based resources too

<u>Image Security</u>

- hosting private image registry/registry is good for security
- kubectl create secret docker-registry <name>
  - specify secret under imagePullSecrets

<u>Security Contexts</u>

- container security settings can be specified in pod level
  - add securityContext under spec in definition file or under container

<u>Network Policy</u>

- PODs can communicate by default with each other
  - Can be prevented with network policies linked to PODs
    - Create with definition file and link with labels

Storage (S8)

Volumes

- In docker volume is linked to container to keep data persistent
- In kubernetes volume is attached to pod to store data
  - Specify volume in spec in definition file and use volume mounts
  - Also possible to specify different type of volumes like aws, gcp, …

Persistent Volumes (PV)

- Cluster wide pool of storage volumes to manage storage more centrally
  - Users can select storage with volume claims
  - Create with a definition file
    - accessMode: Readonlymany, readwriteonce, readwritemany
    - Capacity
    - Volumetype: path, awsStore, etc.
  - *Kubectl get persistentvolume*

Persistent Volume Claims (PVC)

- Volumes getting bound to PVC
  - 1 to 1 releationship
  - If no volume available claim is pending state
- Specified with definition file
  - *Kubectl get persistentvolumeclaim*

Networking (S9)

Switching Routing

- Switching:
    - Switch connects host with interfaces in a network
        - *ip link*
- Routing:
    - Connect networks together
    - Router is also a device in the network
    - Gateway used as path to outside network
        - *route* to display routing table
        - *ip route add <ip> via <ip>*
        - *ip route add default via <ip>*
    - ip forwarding with /proc/sys/net/ipv4/ip_forward
        - /etc/sysctl.conf for persistent forwarding

DNS (Dynamic Name Service)

- Add known hosts to /etc/hosts with ip and hostname (old way)
    - *Hostname* to find out the systems hostname
    - Works for commands like *ping, ssh, curl, …*
- DNS Server resolves the names in modern day infrastructure
    - Add dns server to /etc/resolv.conf
    - Add public nameservers like 8.8.8.8 for internet servers
    - *nslookup <dns-name>* to see server address or *dig*
- Domain Names:
    - Separated with dots to group websites under subdomains
    - *search* entry in /etc/resolv.conf to look up for internal domain names

Network Namespaces

- processes in container have the same namespace
- host has own interfaces, routing tables, arp tables and can have its onw network inside
    - *ip netns add* to add namespace
    - link two namespaces with:
        - *ip link add <veth name1> veth peer name < veth name2>*
        - *ip link set <namespace> netns <namespace1/2>*
        - ip –n <namespace1/2> addr add <ip-addr1/2> dev <veth name1/2>
        - ip –n <namespace1/2> link set <veth name1/2> up
    - to create virtual network between multiple networks, create a virtual switch
        - *ip link add <virtual network name> type bridge*
        - *ip link set dev < virtual network name > up*
        - *ip link add <veth name1> veth peer name < bridge network name1>*
        - *ip link set ip link set <veth name> netns <namespace>*
        - ip link set <bridge network name1> master <virtual network name>
        - ip –n <namespace1/2> addr add <ip-addr1/2> dev <veth name1/2>
        - ip –n <namespace1/2> link set <veth name1/2> up

Docker Networking

- container can be run with network option --network
  - none: no network connection to the outside
  - host: container is connected to host network
  - bridge: internal network is created, where containers are connected to

CNI (Container Network Interface)

- set of standartds defining how programs should be developed to solve container networking
- already has a set of plugins available (BRIDGE, VLAN, IPVLAN, …)
- also third party plugins weaveworks, flannel, etc.
- docker has its own standards

Cluster Networking

- kubernetes applications/parts listen to different ports (see documentation)

Pod Networking

- kubernetes doesn’t have built in solution
  - every pod can reach every pod in same node with a unique ip address
  - every pod should be able to communicate with every pod on other nodes without NAT
- many solutions available like wavenet and flannel
- you can also build your own solution with some shell scripting and virtual networks

CNI in Kubernetes

- see *cni with ps –aux | grep kubelet*   *ls /opt/cni/bin*   and */etc/cni/net.d/10-bridge.conf*

CNI weave works

- uses agents on every node for address resolution and ip address assignment
- can be deployed as service/deamon or as pod
  - kubectl apply –f <weave_url with version>
- kubectl get pods –n kube-system

IP Address Management – Weave

- Manual: host-local plugin for ip address management
  - can be configured in net-script.conf
- weave creates range from 10.32.0.1 to 10.47.255.254 on default

Service Networking

- clusterIP only reachable inside cluster
- exposes port to outside
- kube-proxy creates rules for iptables, ipvs, userspace
  - cat /var/log/kube-proxy.log

DNS in Kubernetes

- deploys built in dns server by default
- url for services: web-service.apps.svc.cluster.local

CoreDNS in Kubernetes

- kubernetes deploys dns server as kube-dns
  - core dns after 12.1
  - deployed as pod's in cluster which runs coredns executable
  - service is also deployed
  - */etc/coredns/Corefile*
  - *Kubectl get configmap*

Ingress

- Layer 7 Load-Balancer built-in kubernetes cluster
- Deploys solution(ingress controller) and specifies rules for configuration(ingress resources)
- Ingress controller:
  - Possible solutions: GCE, nginx, …
  - Deployed as deployment with definition file, needs command to start
  - Pass in configmap object as easy configuration
  - POD_NAME and POD_Namespace as env variables
  - Also needs a NodePort service
  - Also a service account with roles, and rolebindings
- Ingress Resources:
  - Set of rules and configurations for the ingress controller
    - Forwarding, routing, …
  - Created with yaml file as ingress object
  - Kubectl get/describe ingress
  - Configure rules in definition file under spec > rules > http > paths
  - Default backend for wrong routes
  - Use host field to specifie the url

<u>Install Hard Way (S10)</u>

<u>Design a Kubernetes Cluster</u>

- Purpose of Cluster?
    - Education: Minicube or Single Node cluster
    - Development & Testing: Multi Node Cluster with kubeadm
    - Production Application: Multi node cluster with multiple master nodes with kubeadm/GCP/AWS
- Cloud or on Prem?
    - Kubeadm for on prem
    - GKE for GCP, Kops for AWS, AKS for Azure
- Resources:
    - Storage:
        - high performance → SSD
        - multiple connections → network storage
        - also consider persistent storage
    - Nodes:
        - Virtual or physical machines
        - Master and worker nodes depending on workload
        - Not host workloads on master
- How many apps, what kind?
- Traffic amount

<u>Choosing Kubernetes Infrastructure</u>

- Not possible to install natively on windows → always running on Linux
- Minikube for single node cluster
- Kubeadm for single or multi-node cluster (requires VM's)
- Turnkey Solutions:
    - Provision, configure, maintain VM
    - Scripts for deployment
    - Kops on AWS
    - OpenShift: on prem solution with good ci-pipeline possibilities
    - CloudFoundry Container Runtime
    - VMware Cloud PKS
    - Vagrant
    - …
- Hosted Solutions :
    - Kubernetes as a Service
    - Provider provisions, installs and maintains VM's
    - GKE on GCP
    - OpenShift Online
    - Azure Kubernetes Service
    - Amazon Eleastic Container Service
    - …

Choosing a Network Solution

- https://kubernetes.io/docs/concepts/cluster-administration/networking/#how-to-implement-the-kubernetes-networking-model

Configure High Availability

- Consider use of multiple master nodes
- Also reduncancy of all other control plane components
    - Api-server: can be active on all master nodes, node balancer should be used
    - Scheduler and Controller manager: should run in active-standby mode
        - Needs leader elect option in kube-controller manager
    - ETCD: can be on master node or external(safer version)

ETCD in HA

- Distributed database which ensures consistency
    - One node is leader with RAFT protocol
    - Write operation is complete if it is written in quorum(>50%) of nodes
    - Odd number of nodes should be preferred
- Download etcd binary → install → configure service
- etcdctl for commands
    - export ETCDCTL_API=3
    - etcdctl put <key> <value>
    - etcdctl get <key>
    - etcdctl get / --prefix –keys-only (for all keys)
- etcd nodes for HA setup is 3,6 or 7

"Kubernetes the Hard Way" on Virtualbox

- https://github.com/mmumshad/kubernetes-the-hard-way

TLS Bootstrap Worker Nodes

- Certificate operations done automatically with kube-api
- Permissions on master node have to be made
- Needs a bootstrap token for permissions
- System:node-bootstrapper is default role
- Kubectl get csr to vies ertificate requests
- Can be automtically approved by role certificatesigningrequests:nodeclient
- Automatic approvement only for client certificates possible

Install Kubeadm Way (S11)

- https://github.com/BennyTheSen/kubernetes_absolute_beginners_course/blob/master/Kubernetes%20for%20beginners.pdf
- https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

<u>End to End Tests on a Kubernetes Cluster (S14)</u>

<u>End to End Tests</u>

- Manual:
    - Kubectl get nodes
    - Kubectl get pods –n all-namespaces
    - Service …. Status
    - Kubectl run nginx
    - Kubectl scale
    - Kubectl expose …
- Test suite: https://github.com/kubernetes/test-infra
    - Arround 1000 tests spread across 7 different areas (takes ~ 12 hours)
    - Creates namespace → creates object → executes curl on object → records result → cleanup
    - Also good to build and test own kubernetes based solution (needs to pass conformance tests)
- Sonobouy is another tool

<u>End to End Tests – Run and Analyze</u>

- go get –u k8s.io/test-infra/kubetest
- kubetest –extract=<kubernetes-version>
- cd kubernetes
- kubetest –test –provider=<provider> <outfile>
    - skeleton as local provider
    - – test_args to focus on a specific area

<u>Smoke tests</u>

- Demo for cluster installed in S10

<u>Ent to End test part 1</u>

- Demo for cluster installed in S10

Troubleshooting (S13)

Application Failure

- Check accessibility
  - Map your application services and conenctions and check all the parts
  - App: curl
  - Service: kubectl get/describe/logs service
  - Pod: kubectl get/describe/logs pod [--previous]
- https://kubernetes.io/docs/tasks/debug-application-cluster/debug-application/

Control Plane Failure

- Kubectl get nodes
- Kubectl get pods –-all-namespaces
- Service kube-apiserver/kube-controller-manager/kube-scheduler/kubelet/kube-proxy status
- Kubectl logs kube-apiserver

Worker Node Failure

- Kubectl get nodes
- Kubectl describe node
- Top
- Df –h
- Service kubelet status
- Sudo journalctl –u kubelet
- Check certificates

Network Failure

- No content atm

Other Topics (S14)

JSON PATH

- https://kodekloud.com/p/json-path-quiz

Advanced Kubectl Commands

- Kubectl commands work with JSON
  - Kubectl commands can be extended with json path
    - kubectl get pods –o json
    - kubectl get pods –o=jsonpath= ‘{ <query> }’
    - kubectl get pods –o=custom-columns=<Colum name>:<jsonpath>
    - kubect get nodes –sort-by=<json path query>