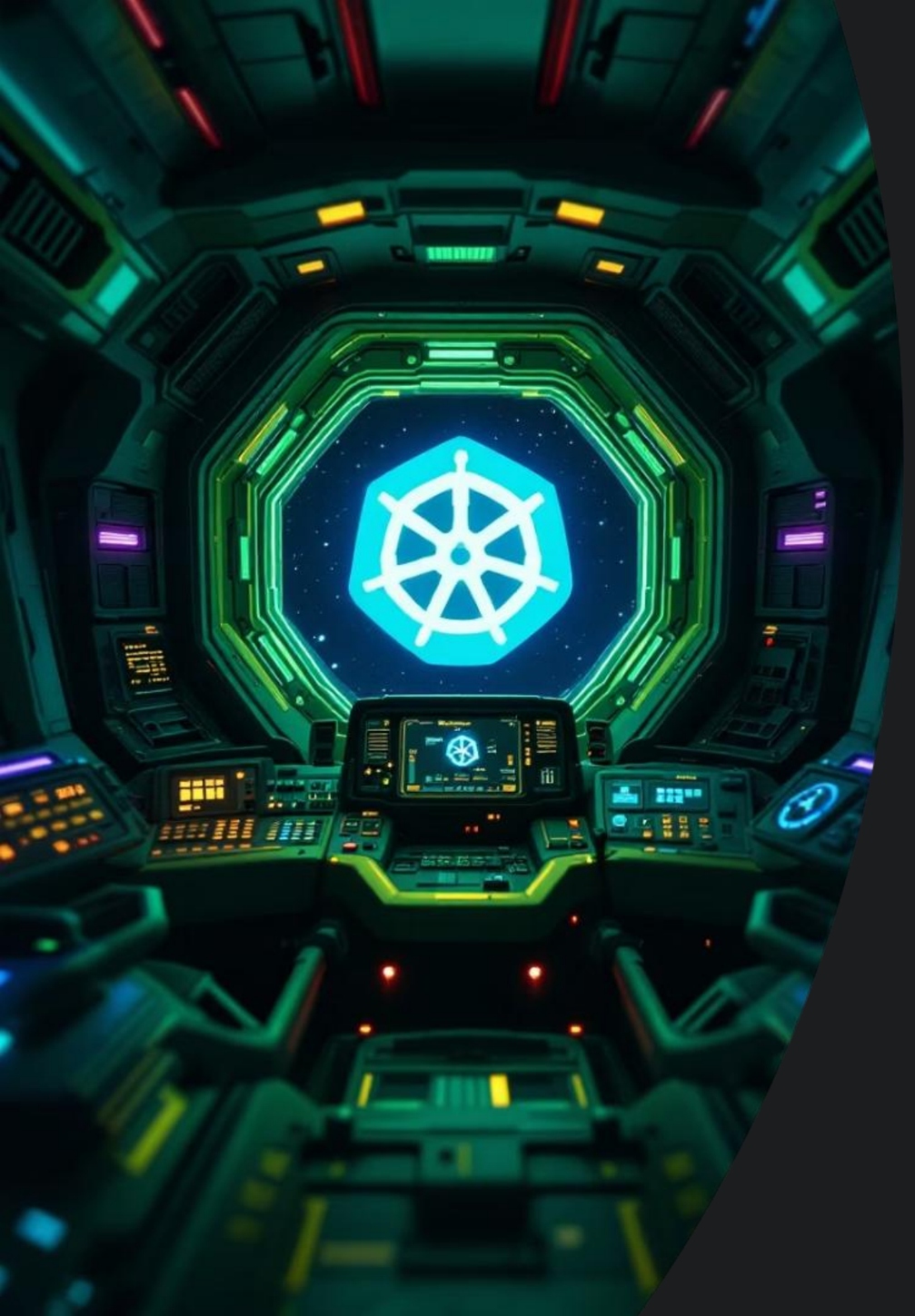


Cloud Fuel

The Kubernetes Saga: From Padawan to Jedi Master

A 5-day journey through the galaxy of container orchestration

May the Force of Kubernetes be with you

xerius

The Kubernetes Galaxy

What is Kubernetes?

Container orchestration platform

Automates deployment, scaling, operations

Why Kubernetes?

Self-healing capabilities

Automatic scaling

Service discovery and load balancing

Understanding Kubernetes: The Imperial Fleet

Control Plane

Imperial command bridge

Nodes

Star Destroyers in formation

Pods

TIE Fighters on patrol

Services

Communication relays

Control Plane components: Imperial Command Bridge

kube-apiserver

Frontend for Kubernetes control plane

Exposes the Kubernetes API

etcd

Consistent, highly-available key-value store

Stores all cluster data

kube-scheduler

Watches for new Pods with no assigned node

Selects nodes for Pods to run on

kube-controller-manager

Runs controller processes

Maintains desired state

cloud-controller-manager

Links cluster to cloud provider API

Node components: Star Destroyer operations

kubelet

Agent that runs on each node

Ensures containers are running in a Pod

kube-proxy

Network proxy on each node

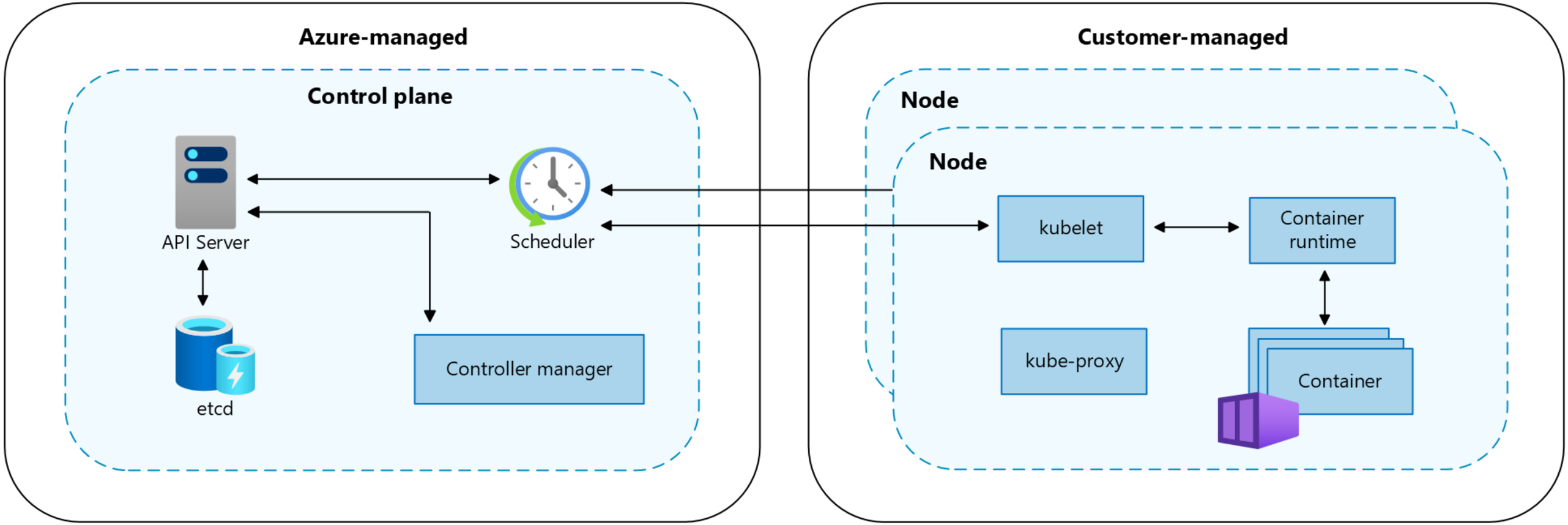
Maintains network rules

Container runtime

Software for running containers

Examples: Docker, containerd, CRI-O

Azure Kubernetes Service: Overview



Imperial Support Fleet: Add-ons

1. **CoreDNS**
   DNS resolution
   Service discovery
2. **Cilium**
   eBPF-powered networking and security
   Advanced network policies and observability
3. **Gateway API**
   Modern ingress and traffic management

Lab 1.00 - Imperial Outpost – Provisioning AKS with Azure CLI

The Emperor commands an Outer Rim digital outpost for tactical systems.

Your mission: provision a robust AKS cluster for Imperial operations.

Mission Objective

Create AKS cluster via Azure CLI

Deploy imperial infrastructure using command line

Configure Workload Identity

Secure access for imperial systems

Enable OIDC issuer

Establish proper authentication protocols

Deploy to designated location

Strategic positioning in France Central

Constructing the Imperial AKS Outpost

Define base parameters

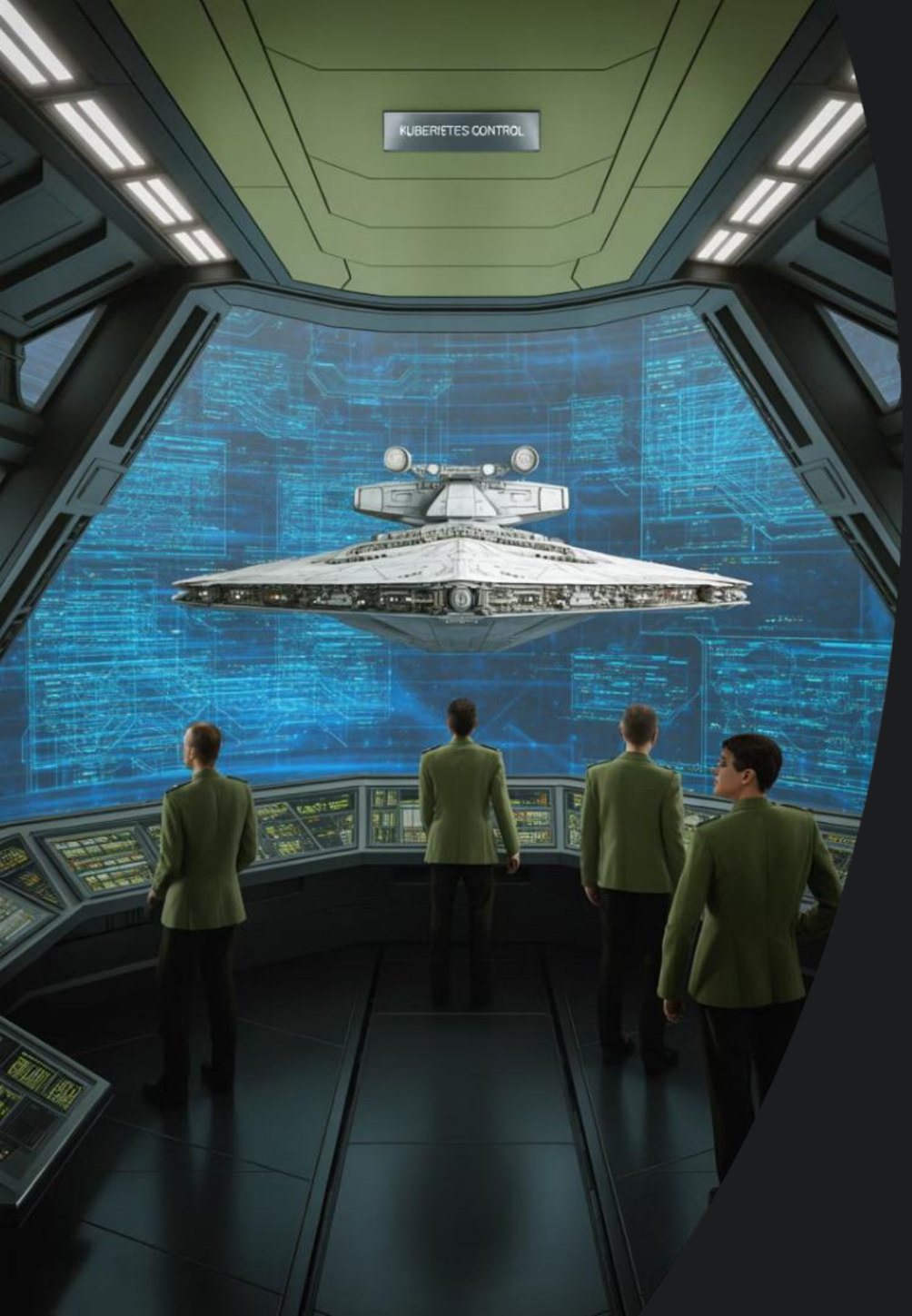
`Set resource group, cluster name, location`

Configure node structure

`Standard_D4ads_v6 size, 2 nodes`

Enable security features

`OIDC issuer, workload identity`



Deploy the AKS Cluster

```
az aks create \
  --resource-group rg-imperial-outpost \
  --name aks-imperial-core \
  --location francecentral \
  --node-osdisk-size 64 \
  --enable-cluster-autoscaler \
  --node-count 1 \
  --min-count 1 \
  --max-count 5 \
  --enable-oidc-issuer \
  --enable-workload-identity \
  --enable-managed-identity \
  --node-vm-size Standard_D4ads_v6 \
  --generate-ssh-keys
```

Execute single command for full deployment

Generates secure access keys automatically

Connect to the Cluster

Get credentials

```
az aks get-credentials --resource-group rg-imperial-outpost --name aks-imperial-core
```

Verify Deployment

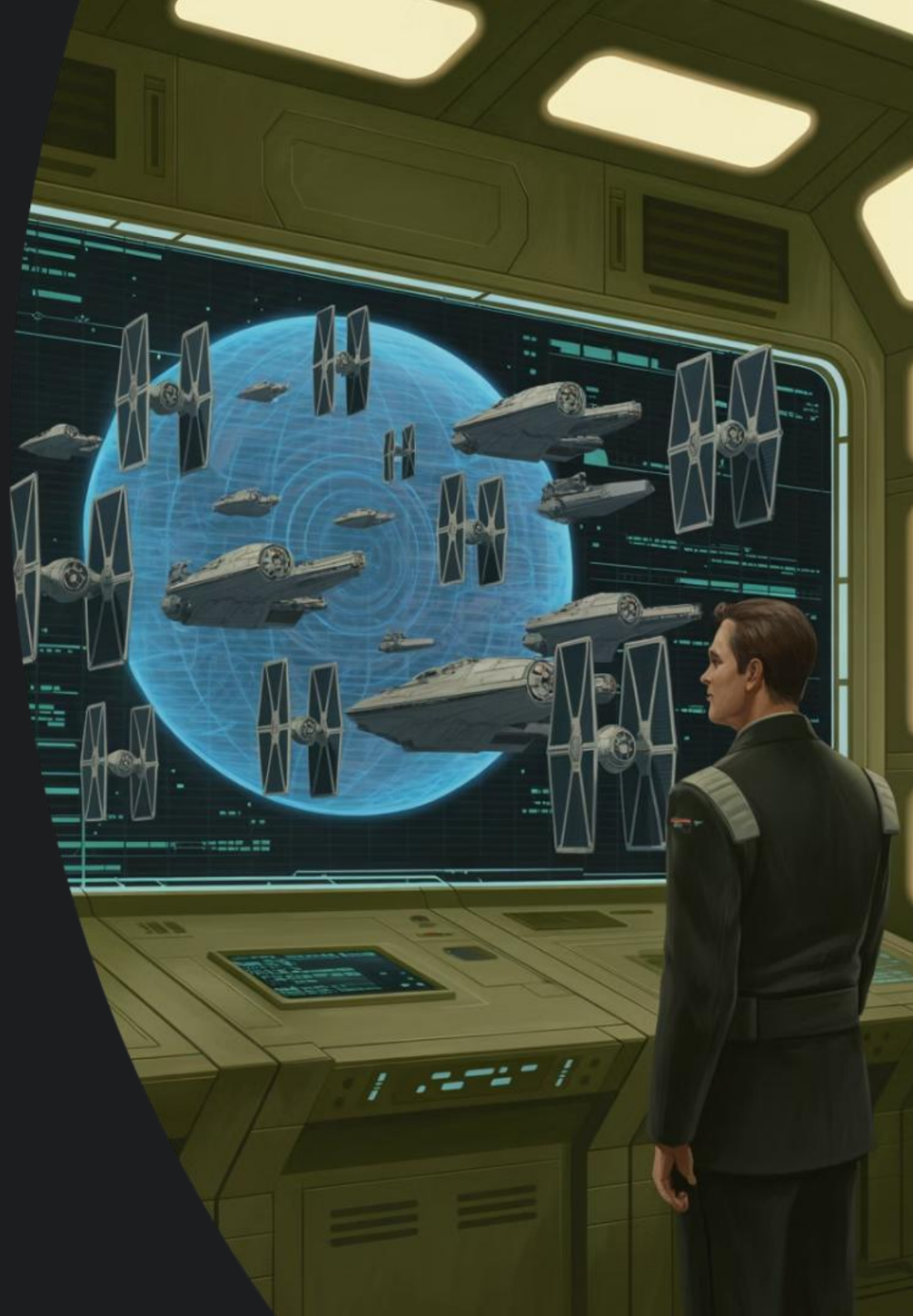
```
kubectl get nodes
```

Begin operations

```
Cluster ready for imperial workloads
```

Imperial Resources: The Tools of Conquest

Your arsenal for galactic domination includes deployments, services, namespaces, and more. Master these resources to command your Kubernetes empire.



Namespaces: Galactic Sectors

Virtual spaces

Divide physical cluster into multiple virtual ones

Access control

Control permissions within namespaces

Resource isolation

Separate resources between teams or projects

The Pod - Your TIE Fighter

Smallest deployable unit

Like a TIE fighter, swift and self-contained

One or more containers

Fighter bay with pilot and systems

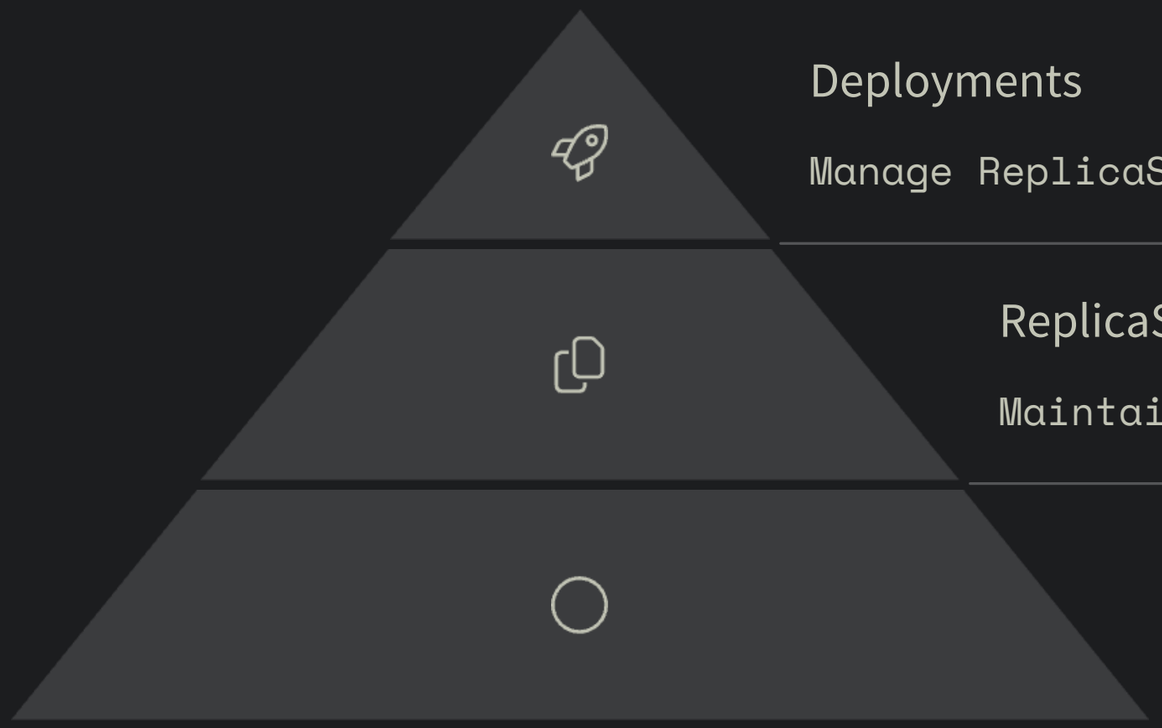
Shared network space

Internal comms within the fighter

Shared storage volumes

Onboard computer and weapons systems

Workload resources: Fleet management



Deployments

Manage ReplicaSets and provide declarative updates

ReplicaSets

Maintain stable set of replica Pods

Pods

Smallest deployable units of computing

Services and networking: Hyperspace routes

Service

Expose applications running on a set of Pods

Endpoints

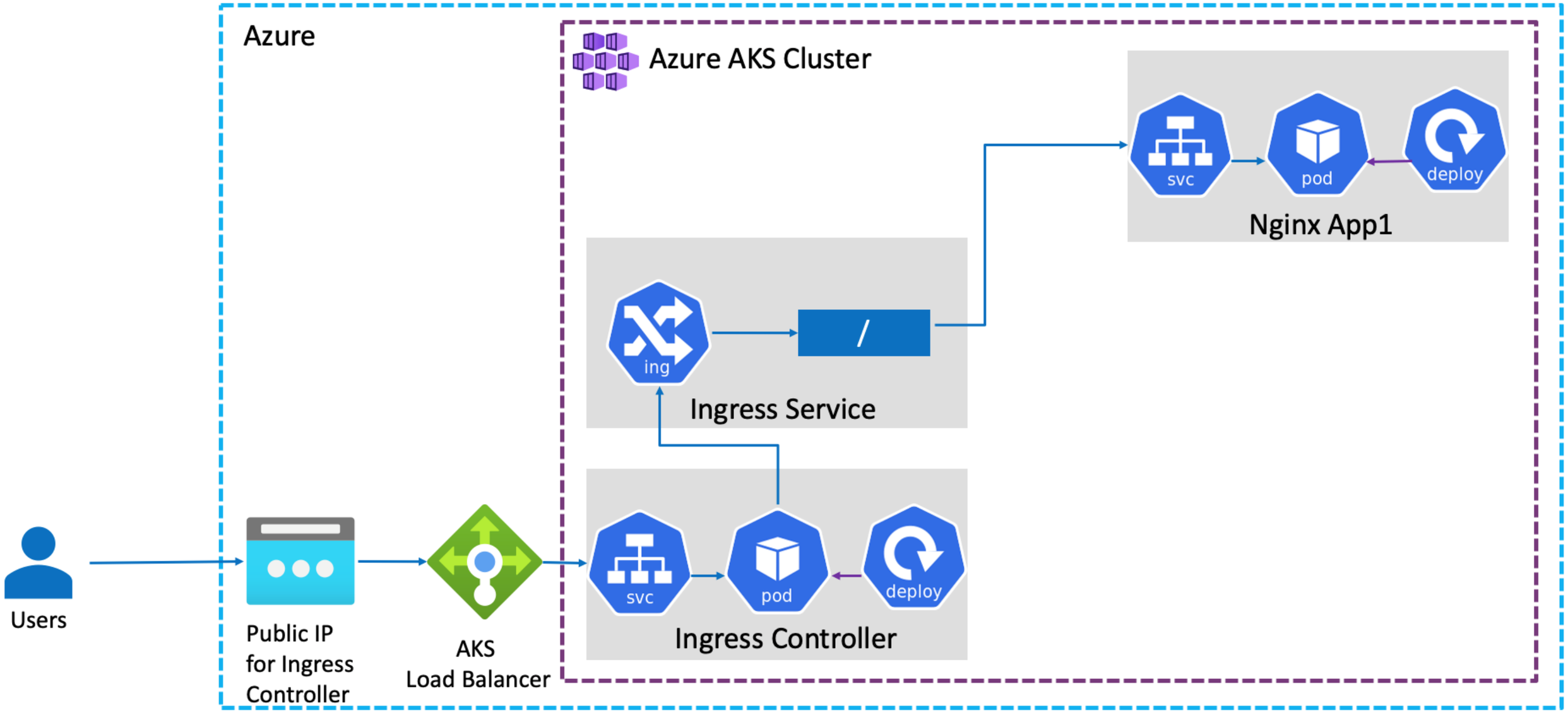
Track Pod IPs that a Service forwards traffic to

Ingress

Manage external access to Services

FQDN: service1.namespace1.svc.cluster.local

Azure AKS & Nginx Ingress – Basic Architecture



Mission agenda: Imperial training protocol

TIE Fighter Pods

Launch single units

Squadron Deployments

Scale multiple fighters

Imperial communications

Services for coordination

Weapon upgrades

ConfigMaps and Secrets

Lab 1.01 - Imperial Deployment: The kubectl strikes back

Master the power of imperative commands without YAML files. Pure terminal domination for rapid imperial victory.

Command & Conquer

Wield imperative kubectl commands to deploy resources instantly from the terminal.

Deploy without delay

Skip YAML manifests and launch pods directly for maximum efficiency.

Monitor your empire

Track deployments across the galaxy with powerful terminal-based tools.

Core deployment commands

Deploy Pods

```
kubectl run stormtrooper --image=nginx:alpine
```

Add Labels

```
kubectl run redis --image=redis:alpine --labels=tier=db
```

Expose Services

```
kubectl expose pod redis --name=redis-service --port=6379
```

Create Deployments

```
kubectl create deployment webapp --image=busybox --replicas=3
```

Advanced Imperial tactics

Custom port configuration

```
kubectl run custom-nginx --image=nginx --port=8080
```

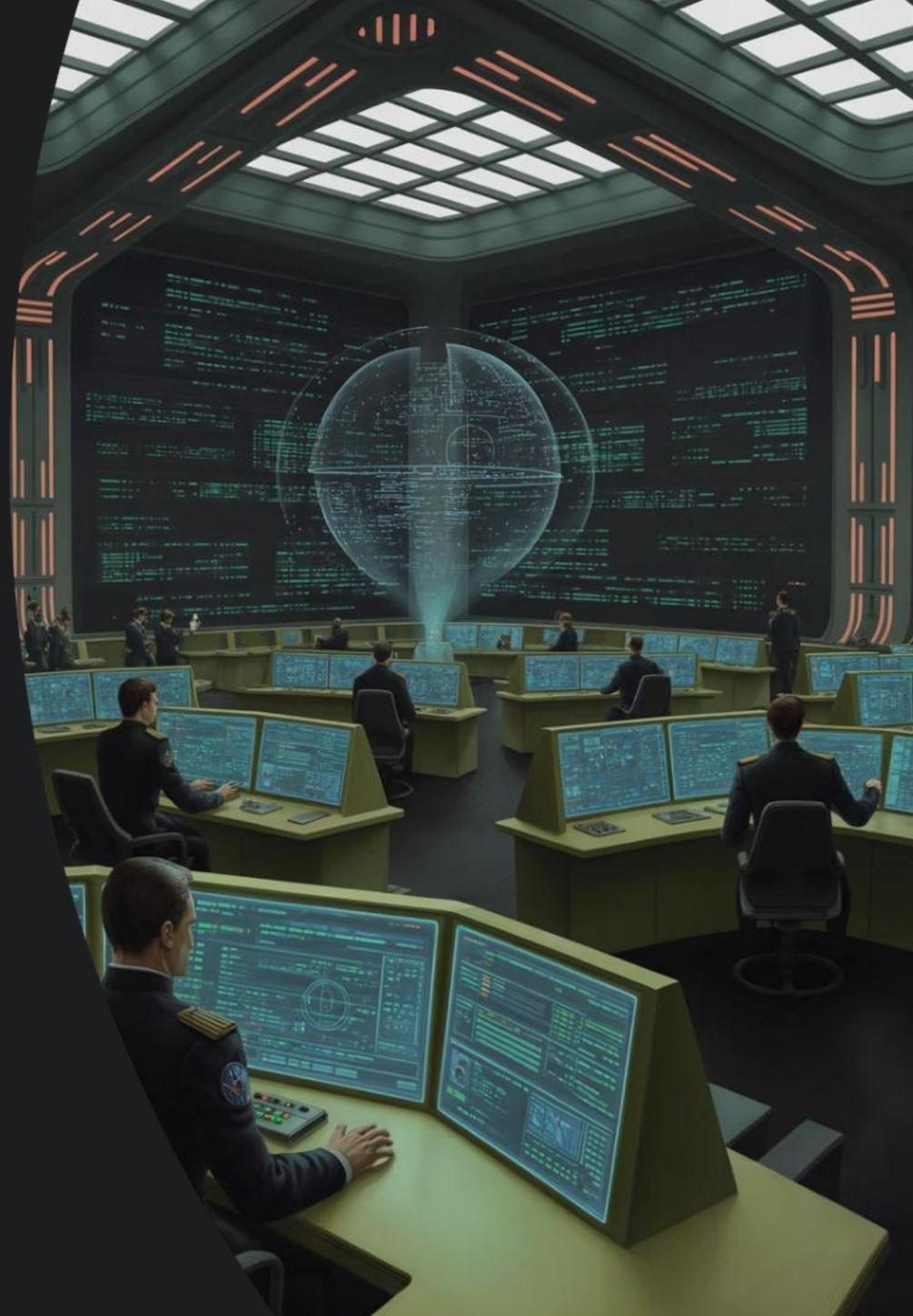
Namespace operations

```
kubectl create namespace dev-ns
```

```
kubectl create deployment redis-deploy --image=redis --replicas=2 -n dev-ns
```

Single command exposure

```
kubectl run httpd --image=httpd:alpine --expose --port=80 --service-name=httpd
```



Imperative vs Declarative: Force Powers

Imperative

- Direct commands
- Quick results
- kubectl run
- Good for testing and troubleshooting

Declarative

- YAML manifests
- Version controlled
- kubectl create/apply
- GitOps friendly

Lab 1.02 - Launching your first TIE Fighter Pod

Define Pod manifest

Create Imperial flight order YAML

Apply to Cluster

`kubectl apply -f tie-fighter.yaml`

Verify launch

`kubectl get pods`

Lab 1.03 - Deployments
TIE Fighter Squadrons

Pod template

TIE fighter blueprint

Replica count

Squadron size order

Self-healing

Auto-replaces damaged fighters

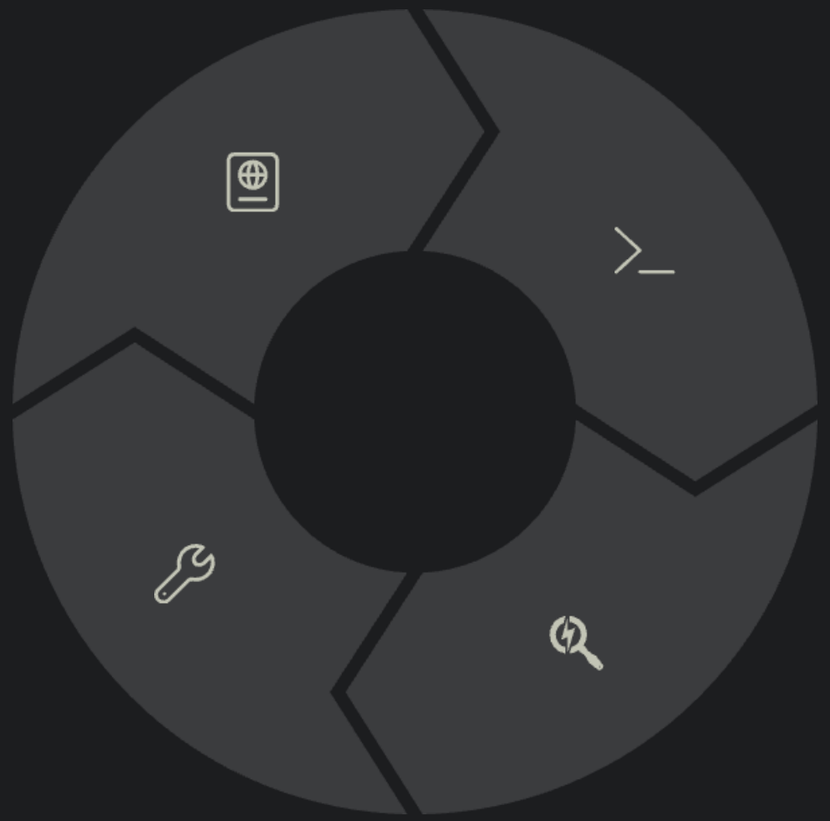
Scaling

Adjust squadron size as needed

Lab 1.03 - Deploying Your TIE Squadron

Create Deployment YAML

Define squadron specs



Apply to Cluster

kubectl apply -f squadron.yaml

Verify Deployment

Check pods and status

Scale as needed

Adjust replica count

Lab 1.03 - Self-healing Imperial auto-repair

Deployment Controller

Constantly monitors pod health

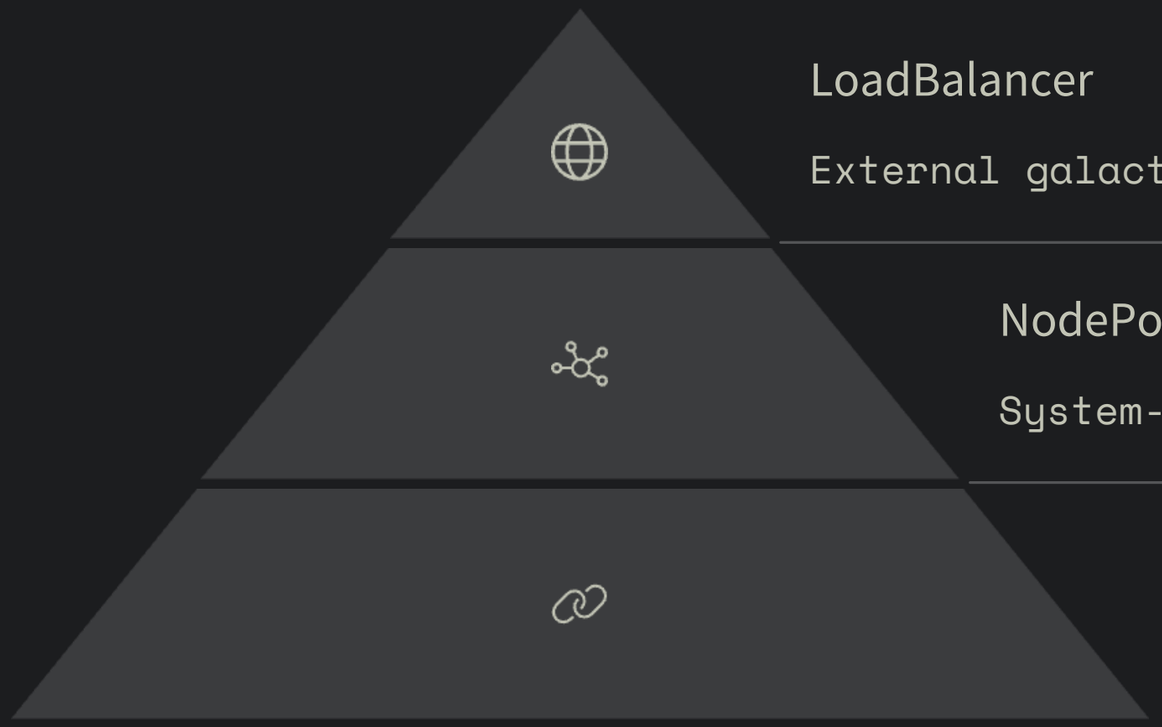
Compares current state with desired state

Initiates repair actions automatically

When a Pod fails:

1. Controller detects failure
2. Old pod marked for termination
3. New pod created from template
4. Squadron strength maintained

Lab 1.04 – Services
Imperial Communications



LoadBalancer

External galactic access

NodePort

System-wide communications

ClusterIP

Internal fleet comms only

Lab 1.04 - Service types

ClusterIP	Internal only	Default type
NodePort	External via node IP	Port 30000-32767
LoadBalancer	External via cloud	Distributes traffic

Lab 1.04 - Creating a communication channel

1. **Define Service YAML**

   Specify service type and port mapping

2. **Apply Service**

   `kubectl apply -f service.yaml`

3. **Verify Connectivity**

   Test comms with curl or browser

Galactic configuration: Imperial Ship settings

Critical configuration parameters ensure your fleet operates at peak efficiency. Master these settings to maintain control across the galaxy.

Imperial Intelligence Archives: ConfigMaps and Secrets

ConfigMaps

Store non-confidential configuration data

Secrets

Store sensitive information like passwords

Mounting

Attach as volumes or environment variables

Updates

Changes propagate to applications

Lab 1.05: ConfigMaps
Tactical Settings

External configuration

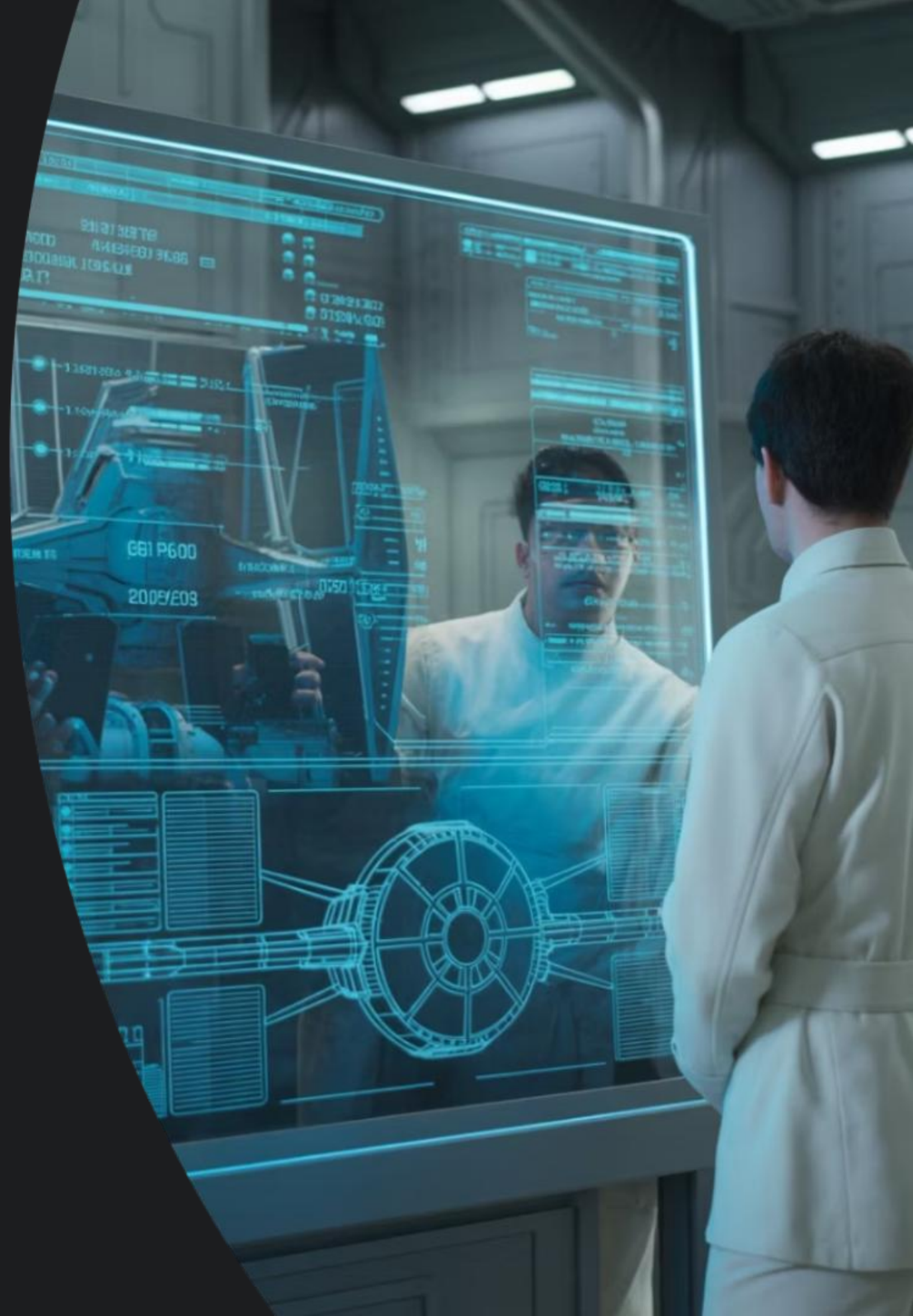
Separate settings from pod specs

Reusable

Apply same config to multiple pods

Updateable

Change settings without rebuilding



Lab 1.05 – Secrets
Imperial Weapon Codes

Sensitive data storage

`Encrypted at rest`

API Keys & passwords

`Base64 encoded`

Certificates & tokens

`Secure injection into pods`

Lab 1.05 – Secrets
Creating weapon system secrets

Create Secret

`kubectl create secret generic tie-weapons`

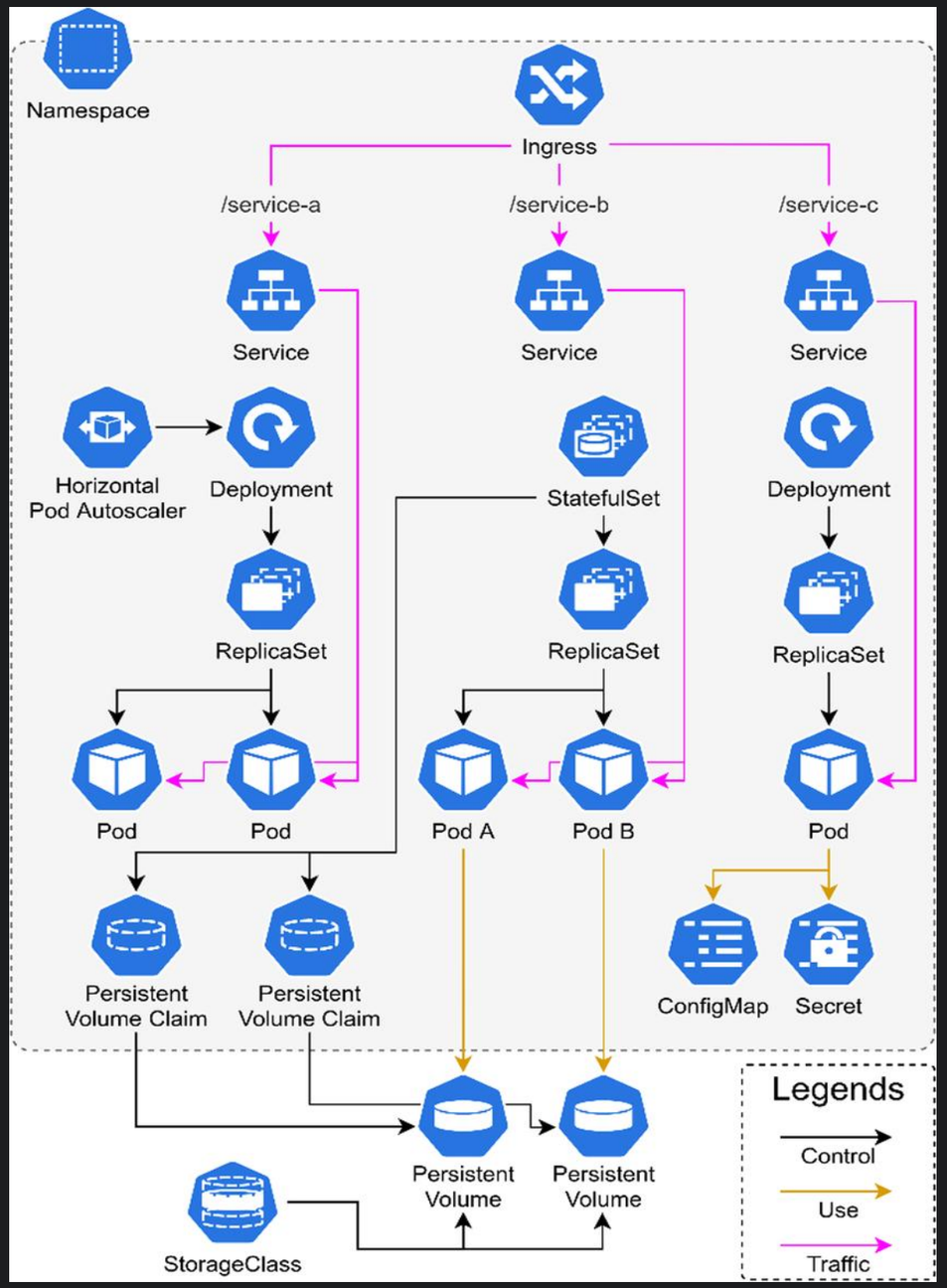
Inject into Pod

`Mount as environment variable`

Verify security

`Check pod has proper credentials`

Resources overview



Lab 1.06: Resource Allocation - Fuel and Power

Squadron Efficiency

Properly allocated resources ensure optimal fleet performance.

Resource Requests

Minimum resources required for TIE fighter operation.

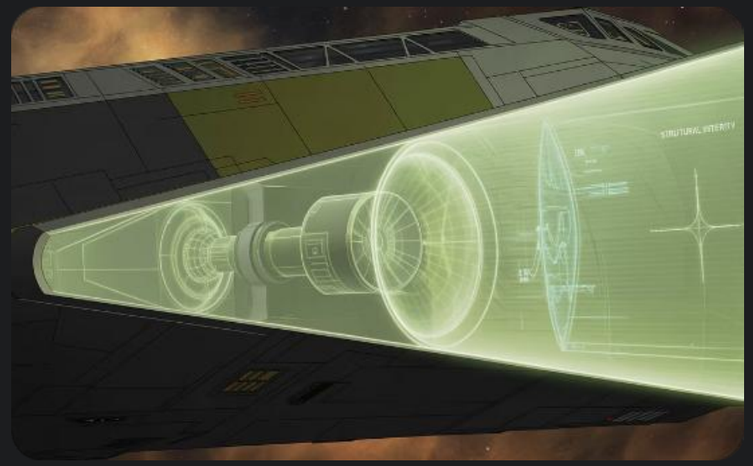
Resource Limits

Maximum resources allocated to prevent system overload.

The Imperial Fleet operates at peak efficiency through precise resource management. Each pod requires specific CPU and memory allocations.

Ship Diagnostics: Probes

Critical systems to monitor your fleet's operational status



Liveness Probe

Determines if a container is running. If it fails, the container is restarted.

Readiness Probe

Checks if a container is ready to receive traffic. Failed probes remove the Pod from service endpoints.

Startup Probe

Indicates when a container application has started. Disables other probes until it succeeds.

Lab 1.07: Diagnostics - Imperial Probe System

Startup Probe

Verifies TIE fighter systems initialize correctly before mission deployment.

- Prevents premature mission assignment
- Checks core systems are operational

Liveness Probe

Continuously monitors vital systems during flight operations.

- Detects frozen navigation systems
- Restarts malfunctioning components

Readiness Probe

Confirms fighter capability to receive imperial communications and orders.

- Prevents routing to unresponsive fighters
- Maintains squadron battle readiness

Persistent Storage in the Galaxy

StorageClass

Defines storage parameters and provisioner

PersistentVolumeClaim

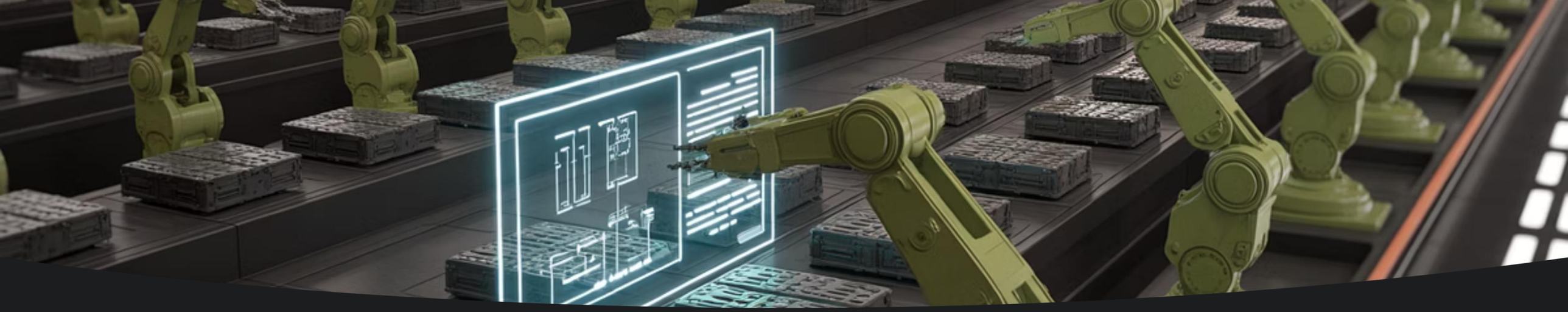
Requests storage and triggers PV creation

Volume Types

CSI, cloud providers, local

PersistentVolume

Automatically provisioned via StorageClass



StorageClass: Dynamic provisioning

Name	Provisioner	Reclaim policy
default	disk.csi.azure.com	Delete
azurefile	file.csi.azure.com	Delete
azureblob	blob.csi.azure.com	Delete

Frequent used storage types in AKS.

Lab 1.08 - Data must survive

Define PersistentVolume

Create 1Gi storage with ReadWriteOnce access mode.

Use hostPath for local testing on imperial systems.

Create PersistentVolumeClaim

Request storage matching your PV specifications.

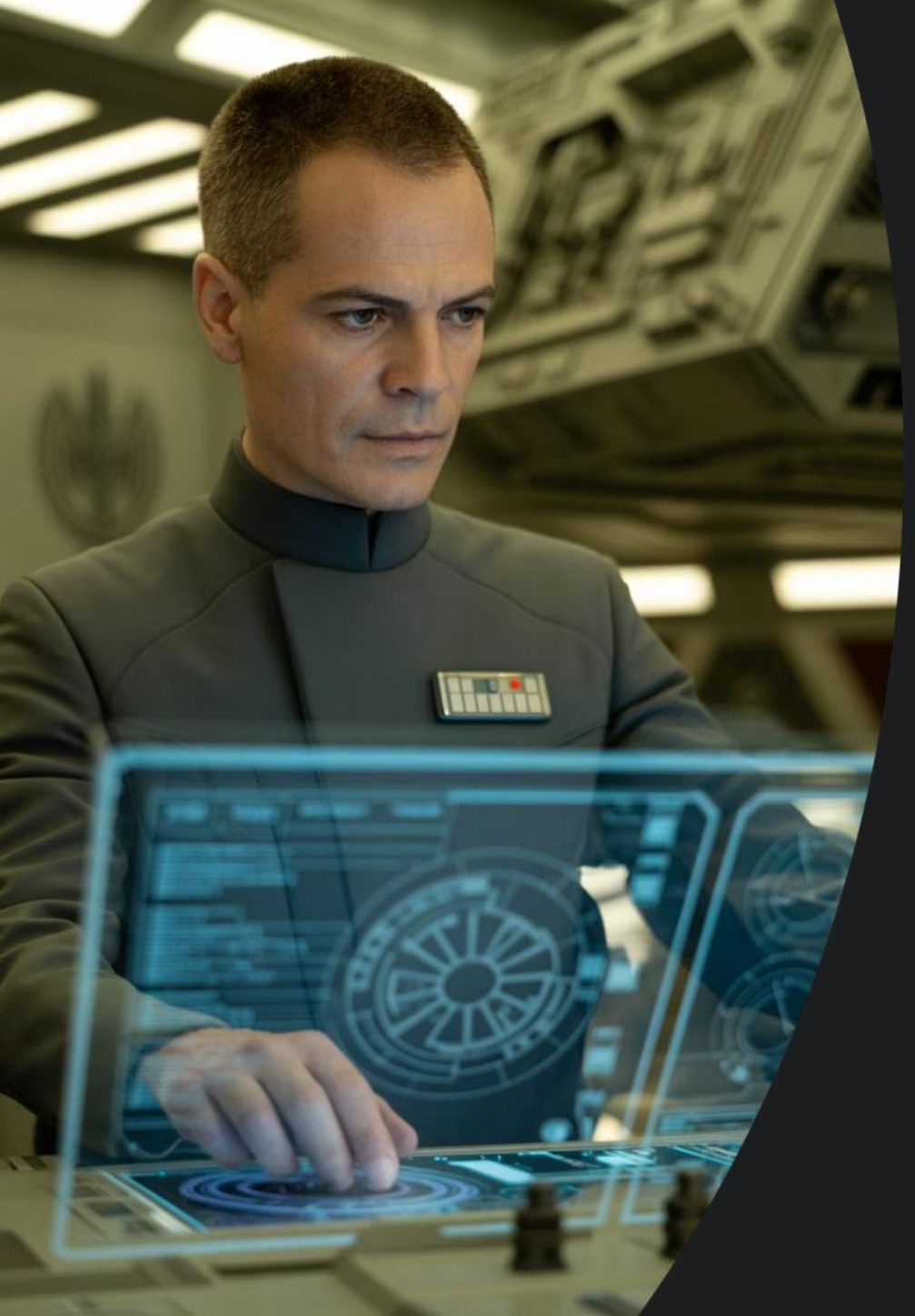
This reserves space for imperial telemetry data.

Deploy Pod with claim

Mount PVC to tie-logger pod at /data path.

Verify data survives pod destruction and recreation.

Advanced mission: Implement dynamic provisioning with Azure Disk CSI for galactic-scale operations.



Lab 1.09: Imperial Command Protocol - Direct Kube API Access

Elite Imperial officers access the Kubernetes API directly for special operations when standard tools are unavailable.

Access Service Account token

Mounted automatically at /var/run/secrets/kubernetes.io/serviceaccount/token

Obtain API Server location

Use KUBERNETES_SERVICE_HOST and KUBERNETES_SERVICE_PORT environment variables

Execute command

Send authenticated curl requests with proper headers and token

The Jedi Knight's Companion: k9s

The Imperial Tool: kubectl

The Empire's command line tool for Kubernetes

The Jedi's Companion: k9s

A powerful terminal UI that lets you command the Kubernetes galaxy with the Force

k9s

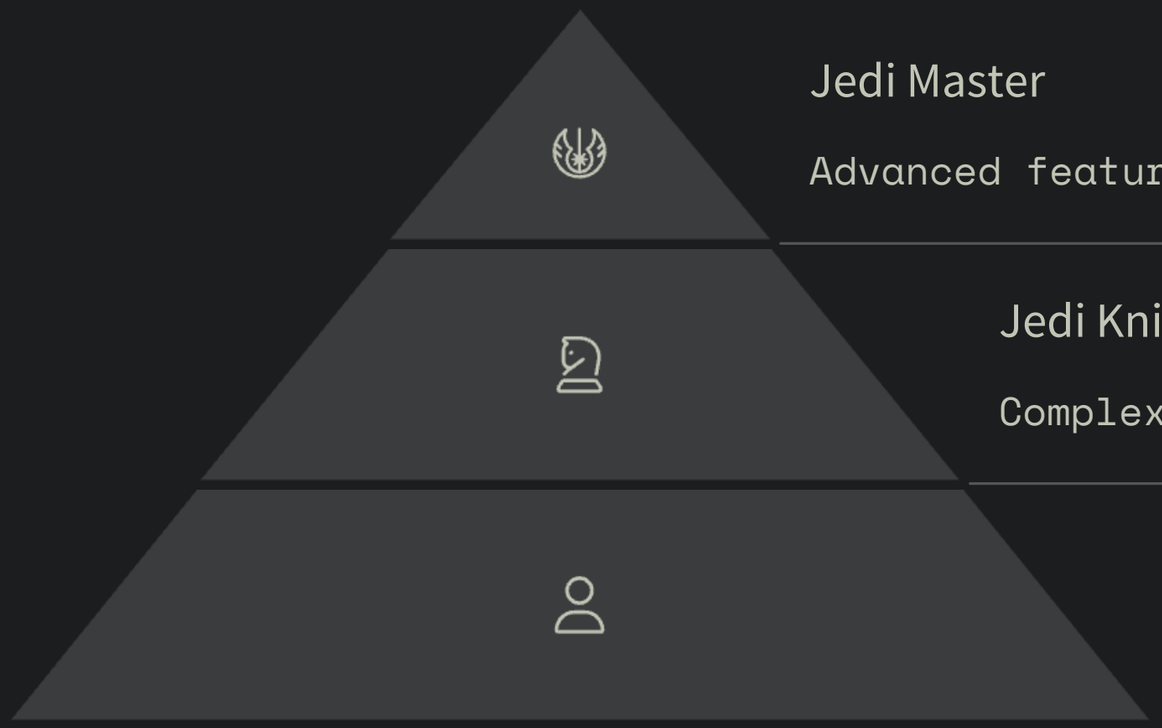
```
Context: aks-imperial-core [RW]          <0> all       <a>       Attach      <ctrl-k>  Kill
Cluster: aks-imperial-core               <1> default   <ctrl-d>  Delete      <l>       Logs
User:    clusterUser_rg-imperial-outpost_aks-impe       <d>       Describe    <p>       Logs Previous
K9s Rev: v0.50.5 ⚡ v0.50.6                             <e>       Edit        <shift-f> Port-Forward
K8s Rev: v1.31.8                                        <?>       Help        <z>       Sanitize
CPU:     2%                                             <shift-j> Jump Owner  <s>       Shell
MEM:     9%
```

pods(all)[33]

NAMESPACE↑	NAME	PF	READY	STATUS	RESTARTS	CPU	MEM	%CPU/R	%CPU/L	%MEM/R	L
default	busybox	●	0/1	Error	0	0	0	n/a	n/a	n/a	a
default	httpd	●	1/1	Running	0	1	4	n/a	n/a	n/a	a
default	nginx	●	1/1	Running	0	0	4	n/a	n/a	n/a	a
default	redis	●	1/1	Running	0	3	9	n/a	n/a	n/a	a
default	squadron-757c875f-bttmm	●	1/1	Running	0	1	4	1	0	7	3
default	squadron-757c875f-frxvd	●	1/1	Running	0	1	4	1	0	6	3
default	squadron-757c875f-wgz2z	●	1/1	Running	0	1	4	1	0	6	3
default	tie-logger	●	1/1	Running	0	1	0	n/a	n/a	n/a	a
default	webapp-85ff4855b-ccmqk	●	0/1	CrashLoopBackOff	45	0	0	n/a	n/a	n/a	a
default	webapp-85ff4855b-mp6k2	●	0/1	CrashLoopBackOff	45	0	0	n/a	n/a	n/a	a
default	webapp-85ff4855b-z9lnm	●	0/1	CrashLoopBackOff	45	0	0	n/a	n/a	n/a	a
kube-system	azure-cns-b5lkr	●	1/1	Running	0	1	35	2	2	14	4
kube-system	azure-cns-tjmpm	●	1/1	Running	0	1	32	2	2	12	2
kube-system	azure-ip-masq-agent-dxspr	●	1/1	Running	0	1	15	1	0	30	6
kube-system	azure-ip-masq-agent-q5tck	●	1/1	Running	0	1	15	1	0	30	6
kube-system	azure-wi-webhook-controller-manager-84fb8c47b7-2nc9r	●	1/1	Running	0	3	14	3	1	73	4
kube-system	azure-wi-webhook-controller-manager-84fb8c47b7-g7rrb	●	1/1	Running	0	3	14	3	1	72	4
kube-system	cloud-node-manager-j6rwc	●	1/1	Running	0	1	16	2	n/a	33	3
kube-system	cloud-node-manager-vs9fk	●	1/1	Running	0	1	14	2	n/a	28	2
kube-system	coredns-57d886c994-jxdrx	●	1/1	Running	0	2	23	2	0	33	4
kube-system	coredns-57d886c994-wgrdq	●	1/1	Running	0	2	25	2	0	35	5
kube-system	coredns-autoscaler-55bcd876cc-mshpp	●	1/1	Running	0	1	11	5	0	113	2
kube-system	csi-azuredisk-node-8djdr	●	3/3	Running	0	3	37	10	n/a	62	0
kube-system	csi-azuredisk-node-zfppk	●	3/3	Running	0	3	49	10	n/a	81	0
kube-system	csi-azurefile-node-6dwl8	●	3/3	Running	0	3	43	10	n/a	73	7
kube-system	csi-azurefile-node-h58wz	●	3/3	Running	0	3	31	10	n/a	52	5
kube-system	konnectivity-agent-autoscaler-679b77b4f-zhvl5	●	1/1	Running	0	1	13	5	0	138	2
kube-system	konnectivity-agent-c8d46697f-dbcsv	●	1/1	Running	0	2	16	10	0	81	1
kube-system	konnectivity-agent-c8d46697f-gdwg5	●	1/1	Running	0	2	17	10	0	87	1

<pod>

Your journey has just begun



Jedi Master

Advanced features and customizations

Jedi Knight

Complex deployments and troubleshooting

Padawan

Well on your journey! Foundations taking shape