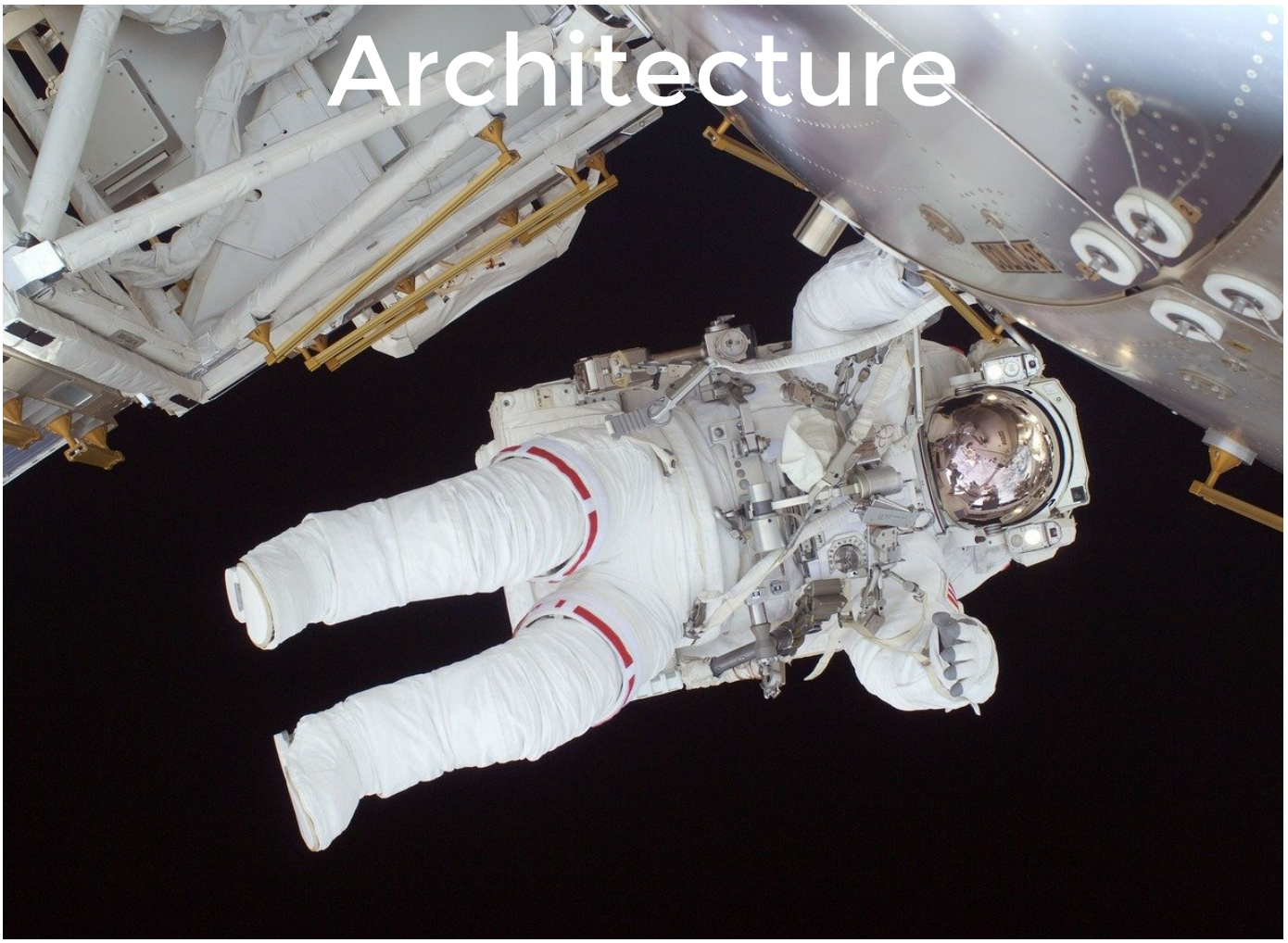
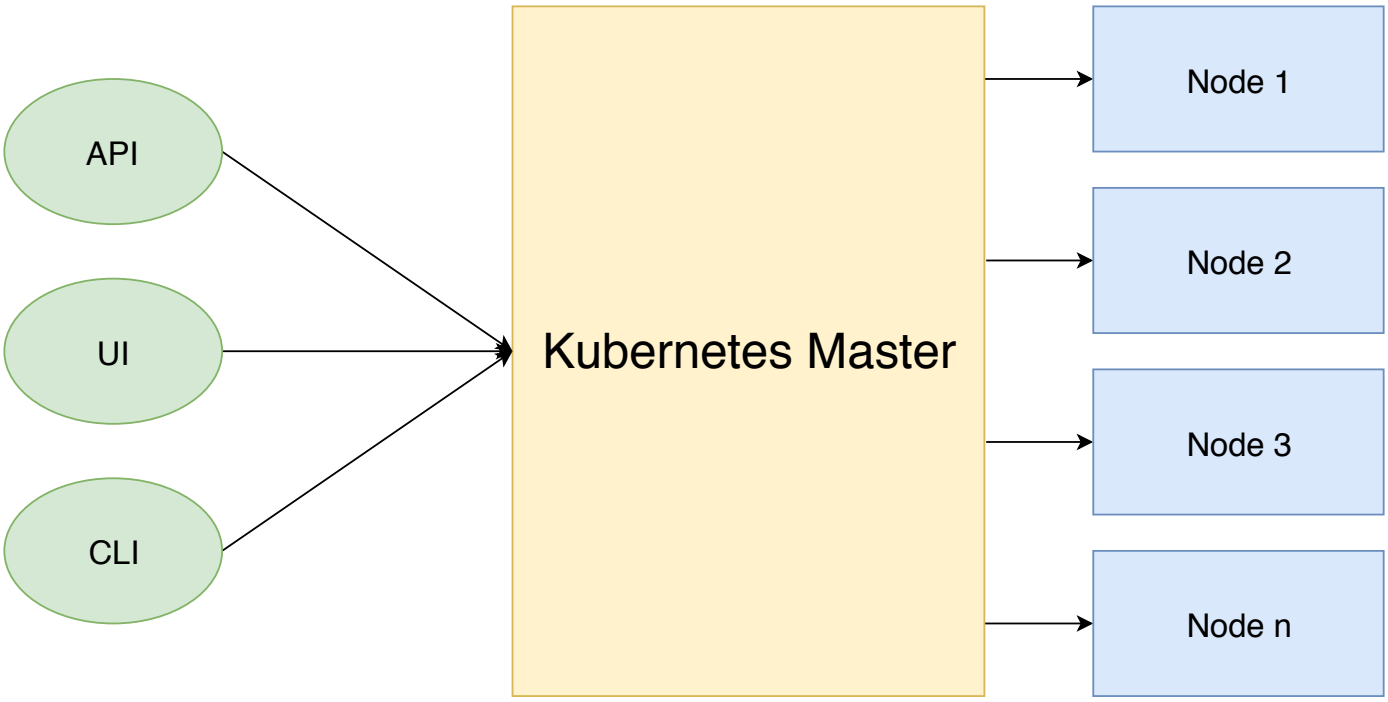
Agenda

Architecture
Organisation (Resources)
Kubernetes Manifests
How to manage a Kubernetes Cluster
First steps (Exercise)
Services and Rollouts (Exercise)
Liveness probe (Exercise)
Comparison to other tools
More Resources
Configuration (Exercise)



What is Kubernetes?

- Open Source (Github)
- Production Container Orchestration
- Automated scaling
- Master-Slave Architecture
- Short k8s


API
UI
CLI
Kubernetes Master
Node 1
Node 2
Node 3
Node n

Organisation

21E06583

Pod 1

- Container 1
- Container 2
- Container 3
- Container n

Pod 2

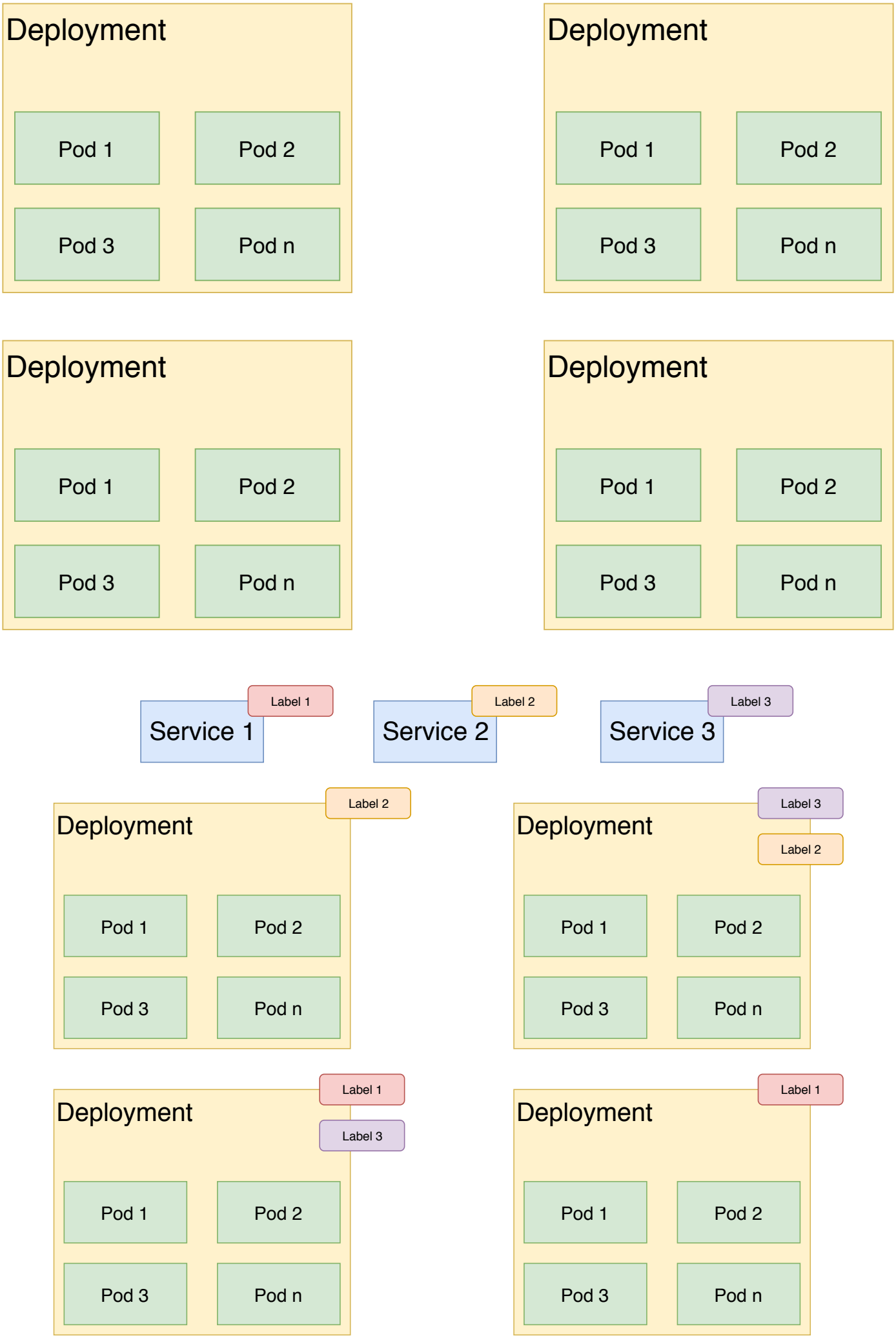
- Container 1
- Container 2
- Container 3
- Container n

Pod 3

- Container 1
- Container 2
- Container 3
- Container n

Pod n

- Container 1
- Container 2
- Container 3
- Container n


Deployment
Pod 1
Pod 2
Pod 3
Pod n
Deployment
Pod 1
Pod 2
Pod 3
Pod n
Deployment
Pod 1
Pod 2
Pod 3
Pod n
Deployment
Pod 1
Pod 2
Pod 3
Pod n
Label 1
Service 1
Label 2
Service 2
Label 3
Service 3
Label 2
Deployment
Pod 1
Pod 2
Pod 3
Pod n
Label 3
Label 2
Deployment
Pod 1
Pod 2
Pod 3
Pod n
Label 1
Label 3
Deployment
Pod 1
Pod 2
Pod 3
Pod n
Label 1
Deployment
Pod 1
Pod 2
Pod 3
Pod n

Kubernetes Manifest

What are Kubernetes Manifests?

- Describe an organisation unit
- Specifies the environment of this unit
- yaml or json syntax

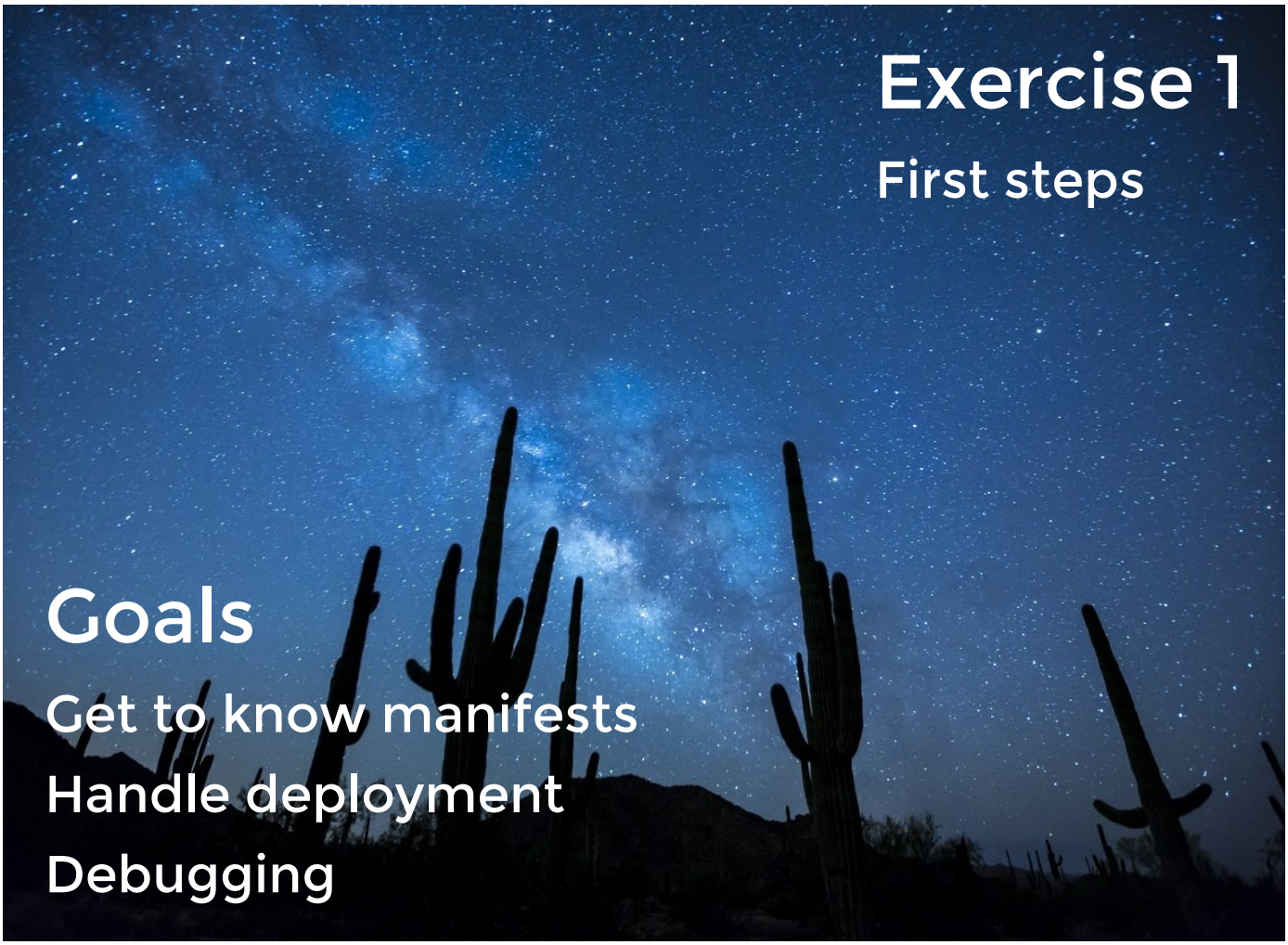
How does a deployment look like?

```yaml
kind: Deployment
metadata:
  name: <name>
spec:
  replicas: <number of replicas>
  strategy:
    <defining rollout strategy>
  template:
    metadata:
      labels:
        app: <label of your app>
    spec:
      containers:
      - name: <container name>
        image: <image with tag>
        imagePullPolicy: Always/IfNotPresent/Never
        ports:
        - containerPort: <port>
          name: <name of the port>
        env:
        - <List of key-value-pairs>
```

How does a Service look like?

```yaml
kind: Service
metadata:
  name: <name>
spec:
  ports:
  - name: <name of the port>
    nodePort: <port number to be exposed to the outside (>30000)>
    port: <port number of the service>
    protocol: TCP
    targetPort: <port number mapped to the containers port>
  selector:
    app: <selector which deployment to choose>
  type: NodePort
```

Kubectl Cheat Sheet



Initial Steps

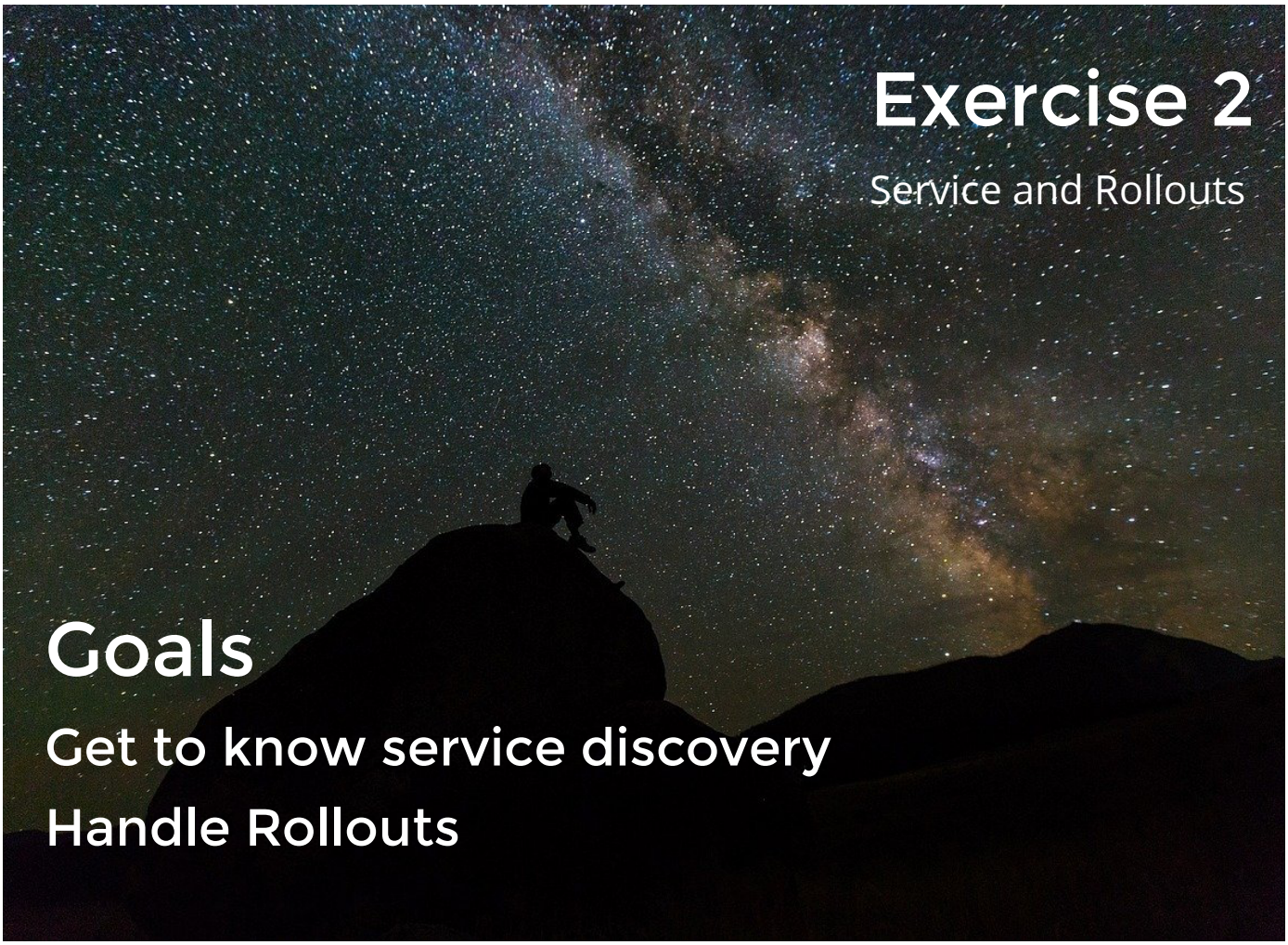
1. Finalise manifest files
2. Create Service and Deployment
3. Try to access the website
4. Retrieve information about the different resources
5. Look at the Pod description

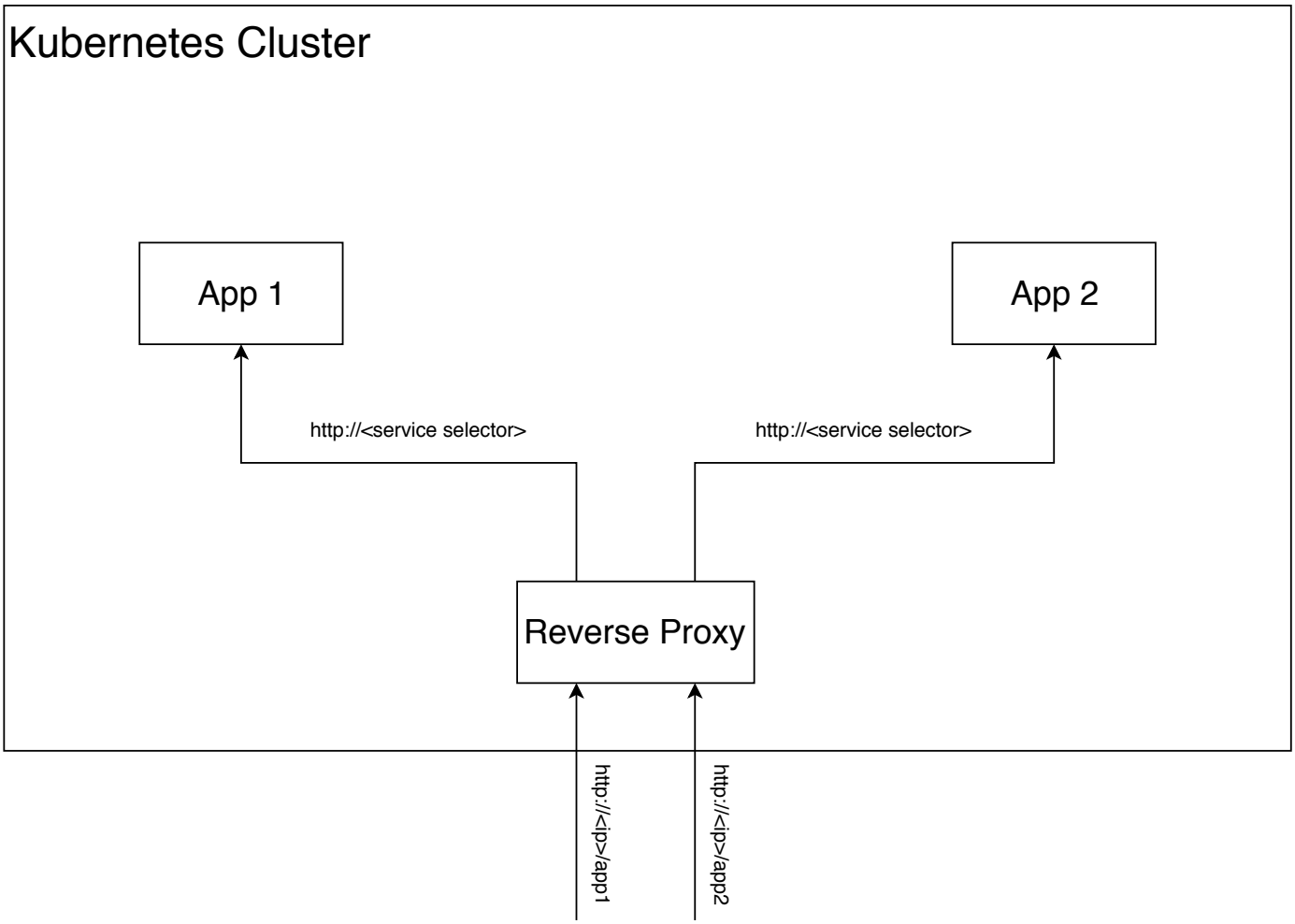
Testing the resilience

1. Delete a Pod and monitor the behaviour
2. Delete the Service and try to access the website
3. Restart the Service
4. Delete the Deployment
5. See what happened to the Pods
6. Restart the Deployment

Accessing Pods

1. Have a look into the logs of a single Pod
2. Run a 'ls' command in a pod
3. Step into the Pod and stop the running process
4. Have a look on the Pods
5. Edit the Deployment on command line
6. Forward the port 80 of a Pod to 8080
7. Try to access http://localhost:8080


Exercise 2
Service and Rollouts
Goals
Get to know service discovery
Handle Rollouts

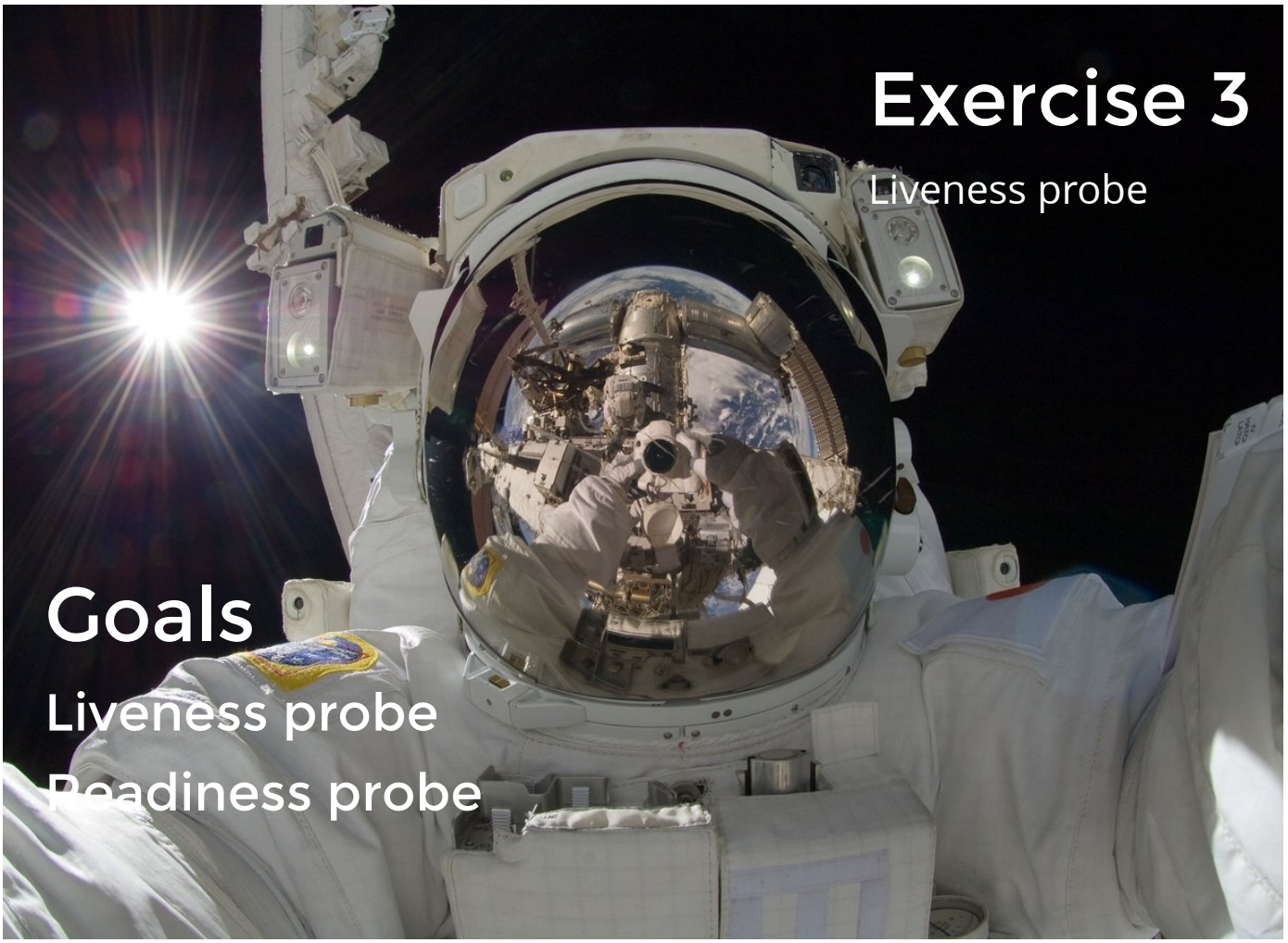

Kubernetes Cluster
App 1
App 2
http://<service selector>
http://<service selector>
Reverse Proxy
http://<ip>/app1
http://<ip>/app2

Service discovery

1. Finalize manifest files
2. Create Service and Deployment
3. Try to access the website (/app1 || /app2)
4. Step into Reverse-Proxy && install curl
5. Run curl http://flask-test-app-1/
6. Exit the Pod

Rollouts

1. Update image of a App Deployment an invalid value
2. Update the Deployment (use replace -f)
3. Monitor the Pods & Deployments
4. Watch Pod Error Message
5. Rollback the Deployment (kubectl undo deployment)



Adding liveness probe

1. Add liveness & readiness probe to App Deployments
   - Liveness probe: /system/info
   - Readiness probe: /system/health
2. Deploy Service and Deployments
3. Is there a difference between the App and Reverse-Proxy Deployment resource?

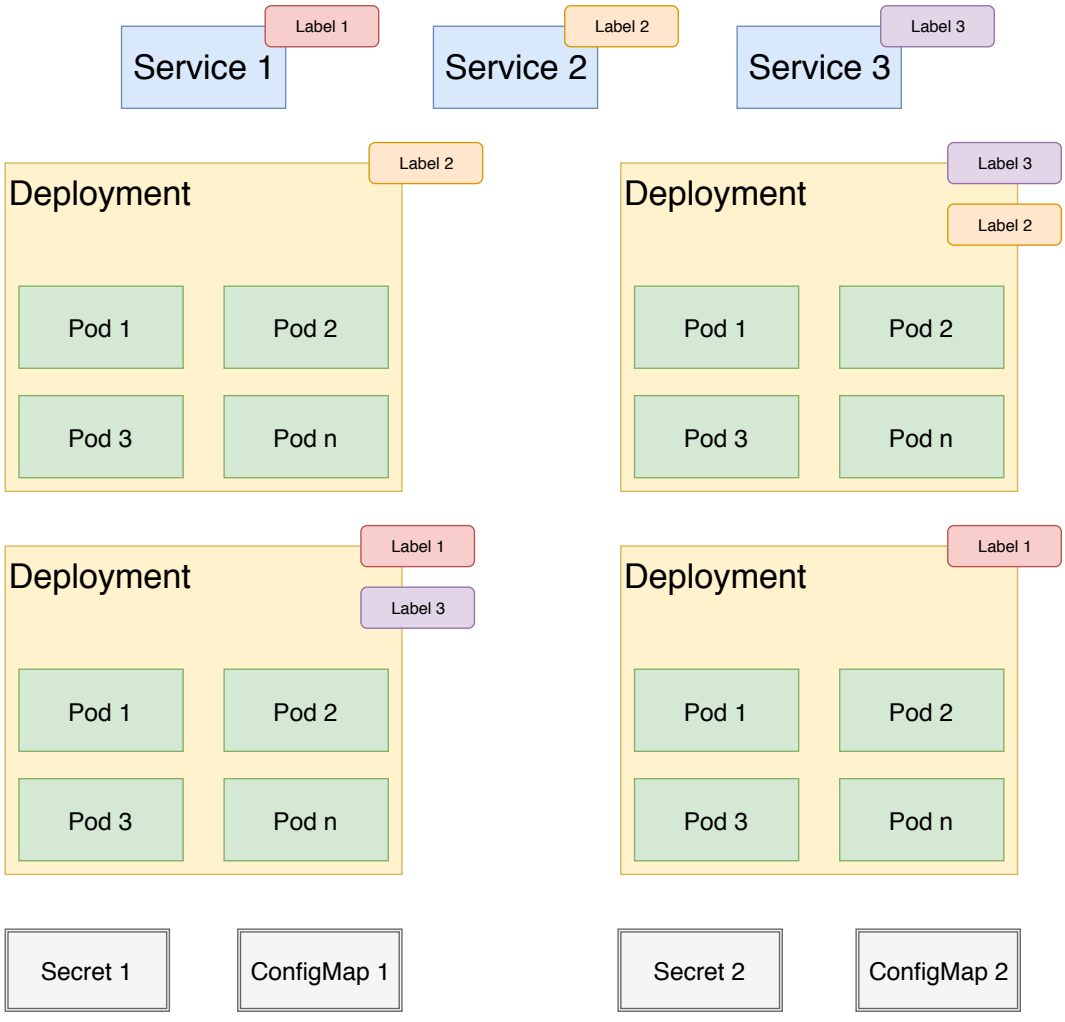
Testing liveness probe

1. Change one Deployment to have non-working probes
2. Watch the behaviour of the pods

Comparison to other tools

	Swarm	Kubernetes	Mesos
Feature Set	−	+	+ −
Extensions	−	−	+
Simple Setup	+	−	−
Fast Learning	+	−	−
Load Balancing	+ −	+	−
Rollouts	+ −	+	+
Health Check	+	+	+
Large Production Deployments	−	+	+
Summary	Simple webapps	greenfield approach	already running legacy

More Resources

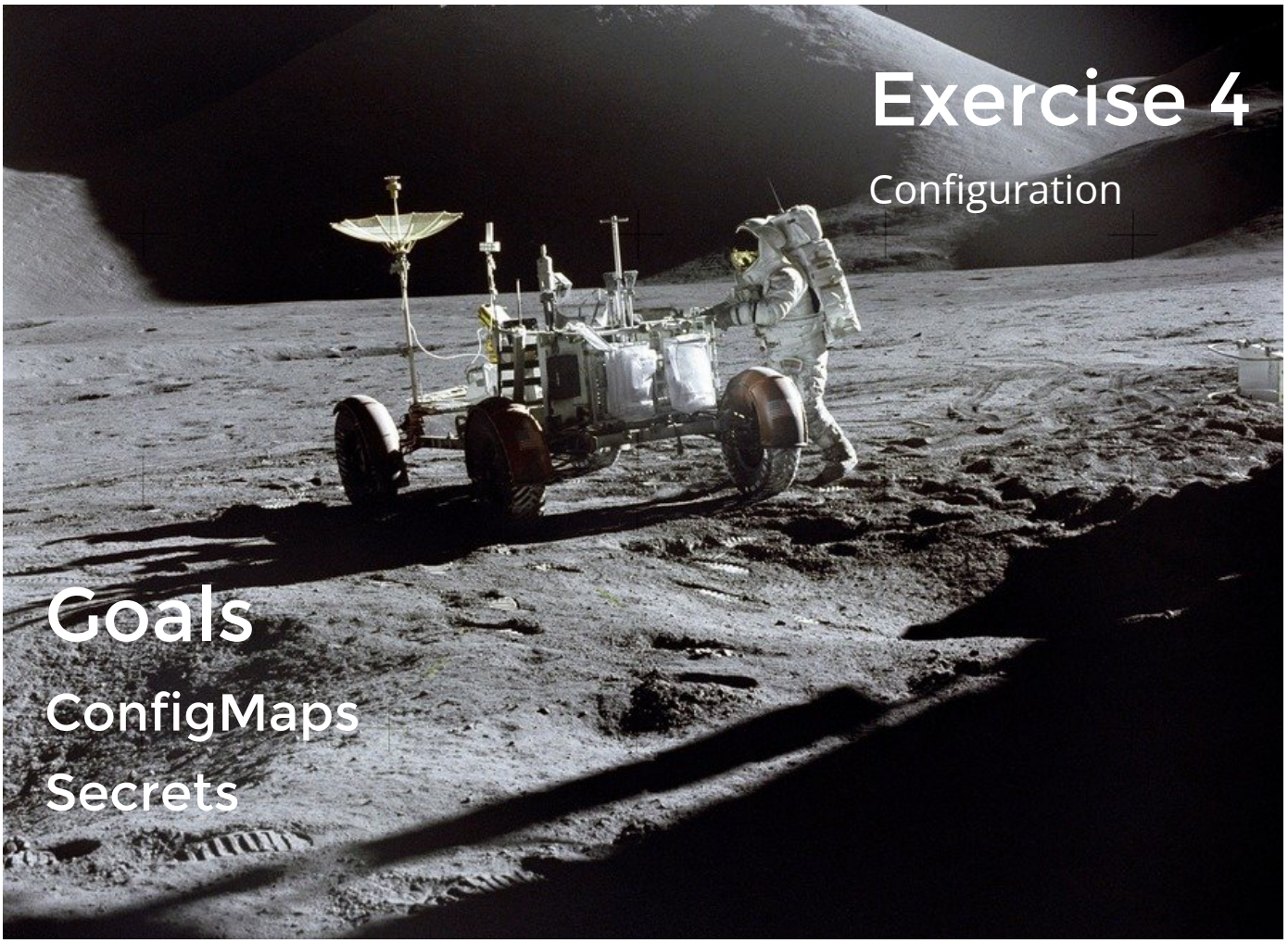


ConfigMap

```
apiVersion: extensions/v1beta1
kind: ConfigMap
metadata:
  name: <name>
apiVersion: v1
data:
  key: <value>
```

Secret

```
apiVersion: extensions/v1beta1
kind: Secret
metadata:
  name: <name>
apiVersion: v1
data:
  key: <value>
```



Configuration Management

1. Use a configMap to configure environment variables
    1. Check if the variable is correctly injected
2. Use a secret to configure environment variables
    1. Delete the configMap beforehand
    2. Remember to encode the variable with base64
    3. Check if the variable is correctly injected
3. Delete the secret
4. Create the secret directly from the command line

Where are the differences?

Summary
Professional Training
Day 3