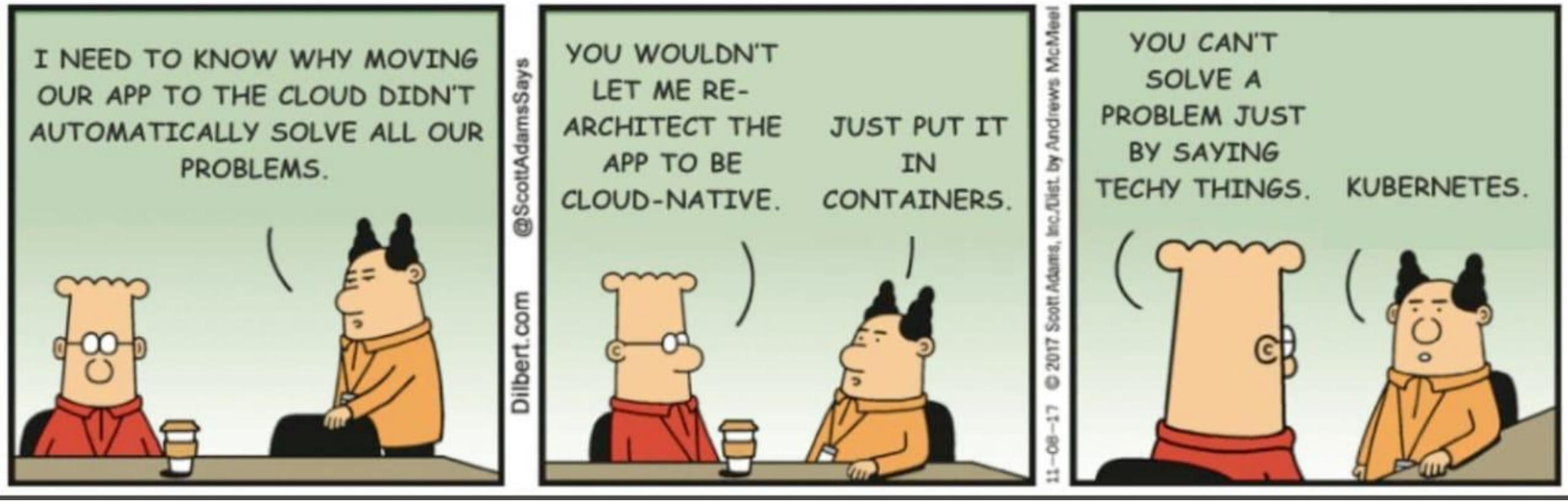
Lecture 13 - Tools

General matters

- Very little feedback on exam arrived
- I will now start preparation of exam for 14. - 23.12
- Another opportunity in ~February
- Sign in with Sisu
- Unless, new Covid-19 becomes worse, we use normal exam rooms
- For those reported not being in Finland I will organize special exam

Today

- Kubernetes
- Ansible
- Cloud Foundry & Heroku


I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.
@ScottAdamsSays
Dilbert.com
11-06-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel

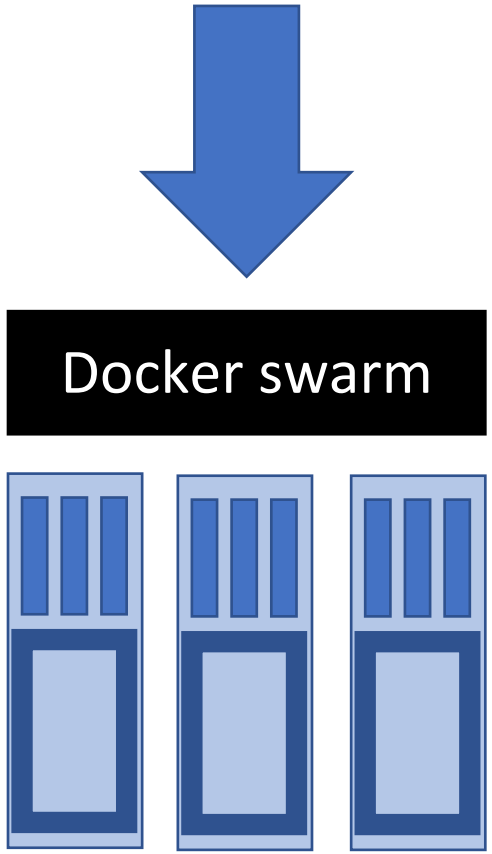
What is ”cloud orchestration”?

Two results of googling

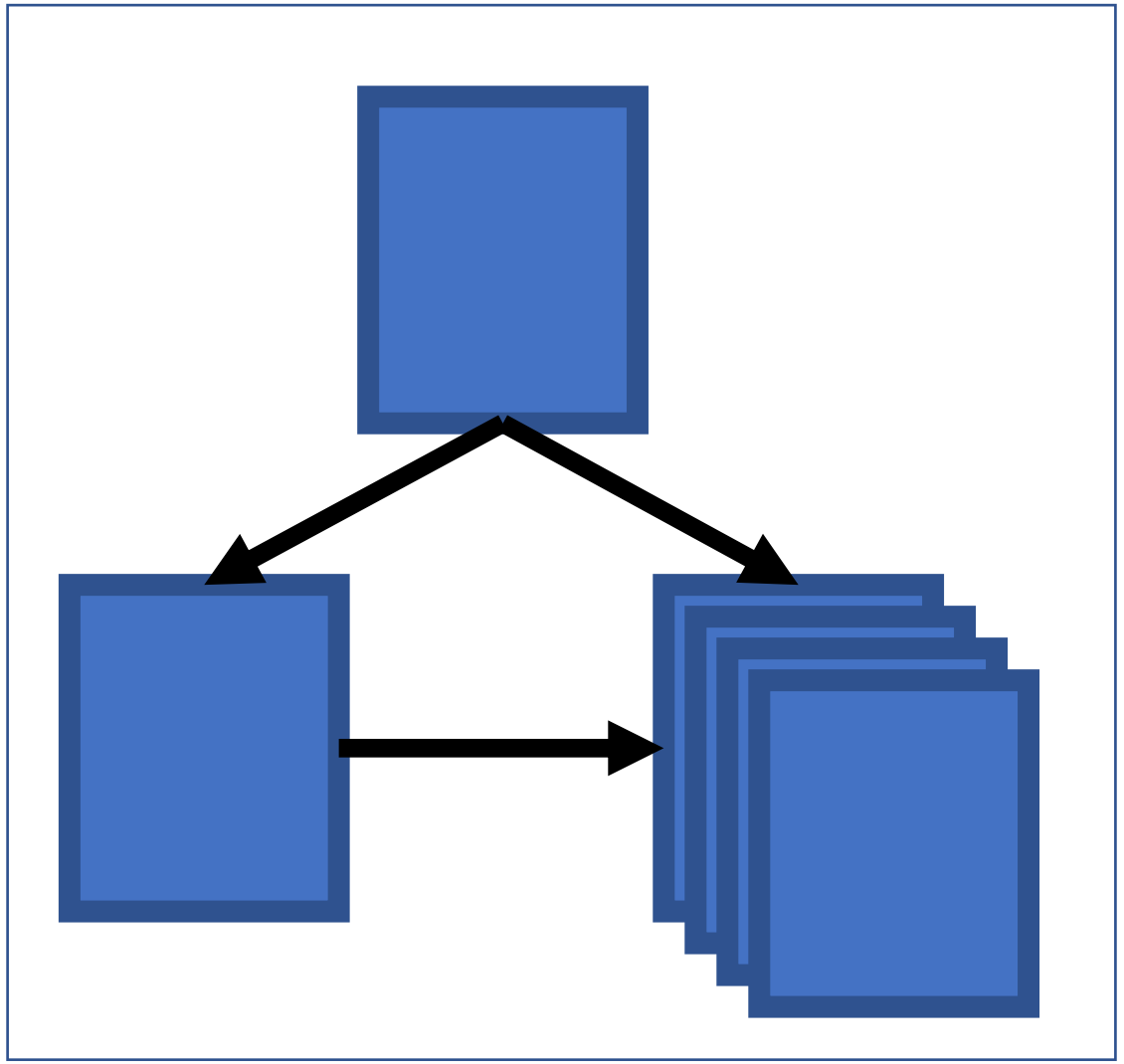
- **Orchestration** is the automated configuration, coordination, and management of computer systems and software
- Cloud orchestration is the use of programming technology to manage the interconnections and interactions among workloads on public and private cloud infrastructure. It connects automated tasks into a cohesive workflow to accomplish a goal, with permissions oversight and policy enforcement.

Docker swarm - docker compose

Orchestration



Docket compose



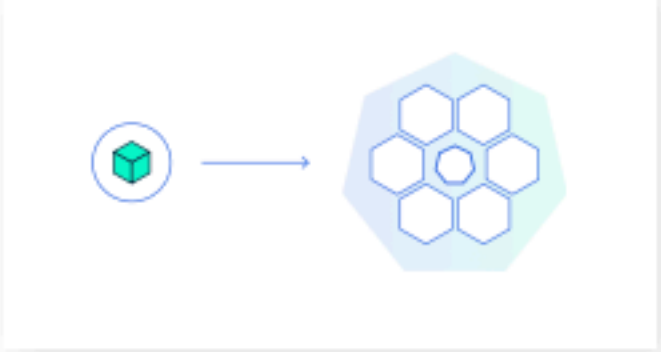
Kubernetes (k8s)

- An advanced tool for orchestration of containers (or other similar things)
- Comes with several features supporting automation and monitoring
- Has as steep learning curve
- Open source released by Google in 2004 (apache license)
- Now maintained by Cloud Native Computing Foundation (https://www.cncf.io)
- "platform for automating deployment, scaling, and operations of application containers across clusters of hosts”
- Is “job interview stuff”

Modules of kubernetes.io tutorial

(source: https://kubernetes.io/docs/tutorials/kubernetes-basics/)

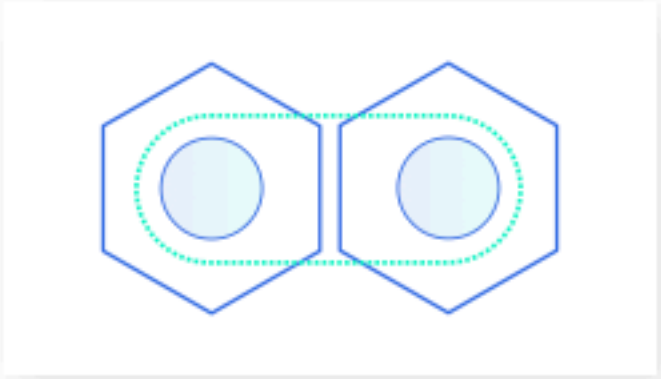
1. Create a Kubernetes cluster



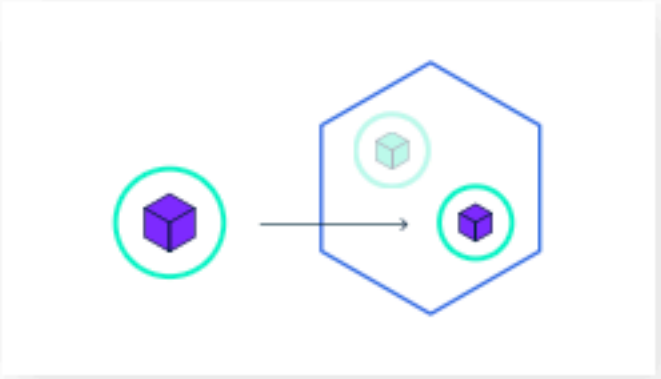
2. Deploy an app

3. Explore your app

4. Expose your app publicly

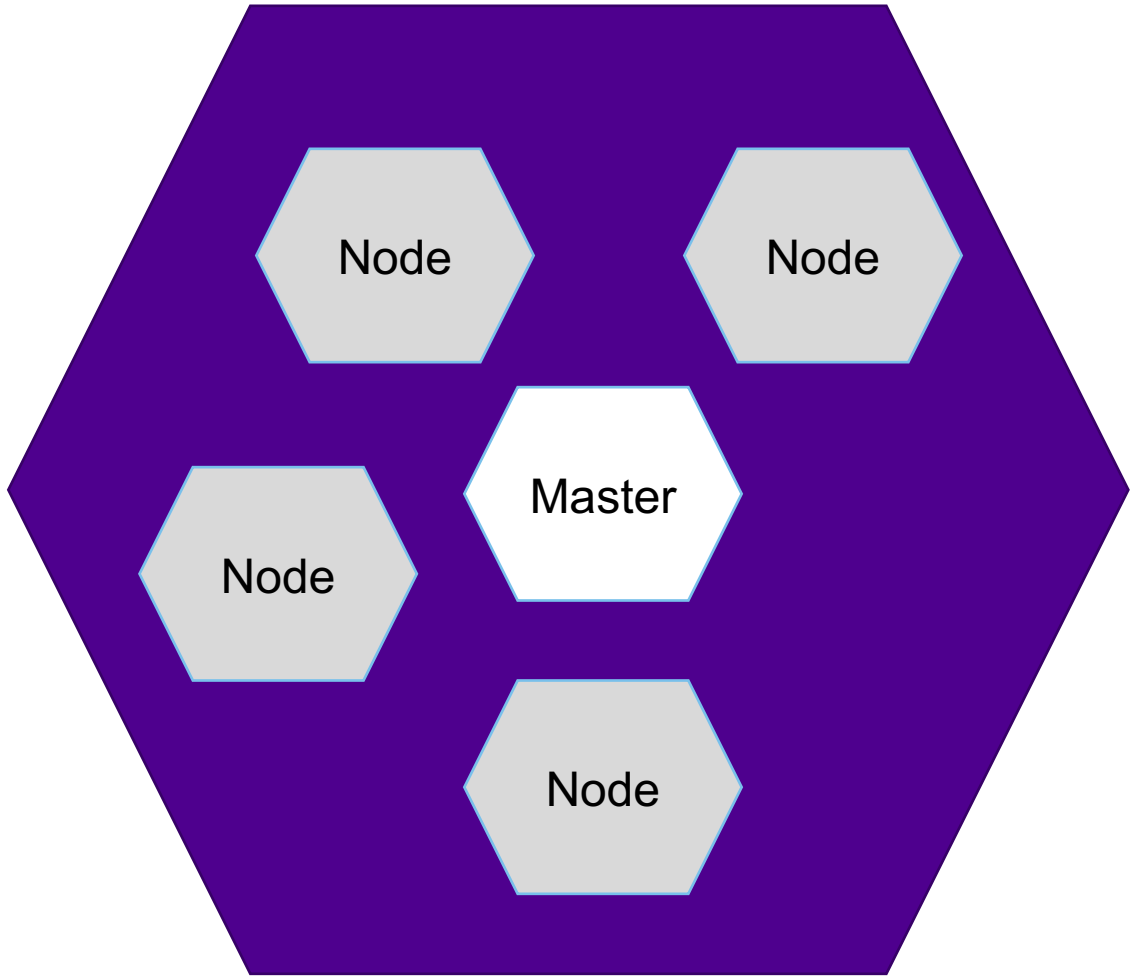


5. Scale up your app

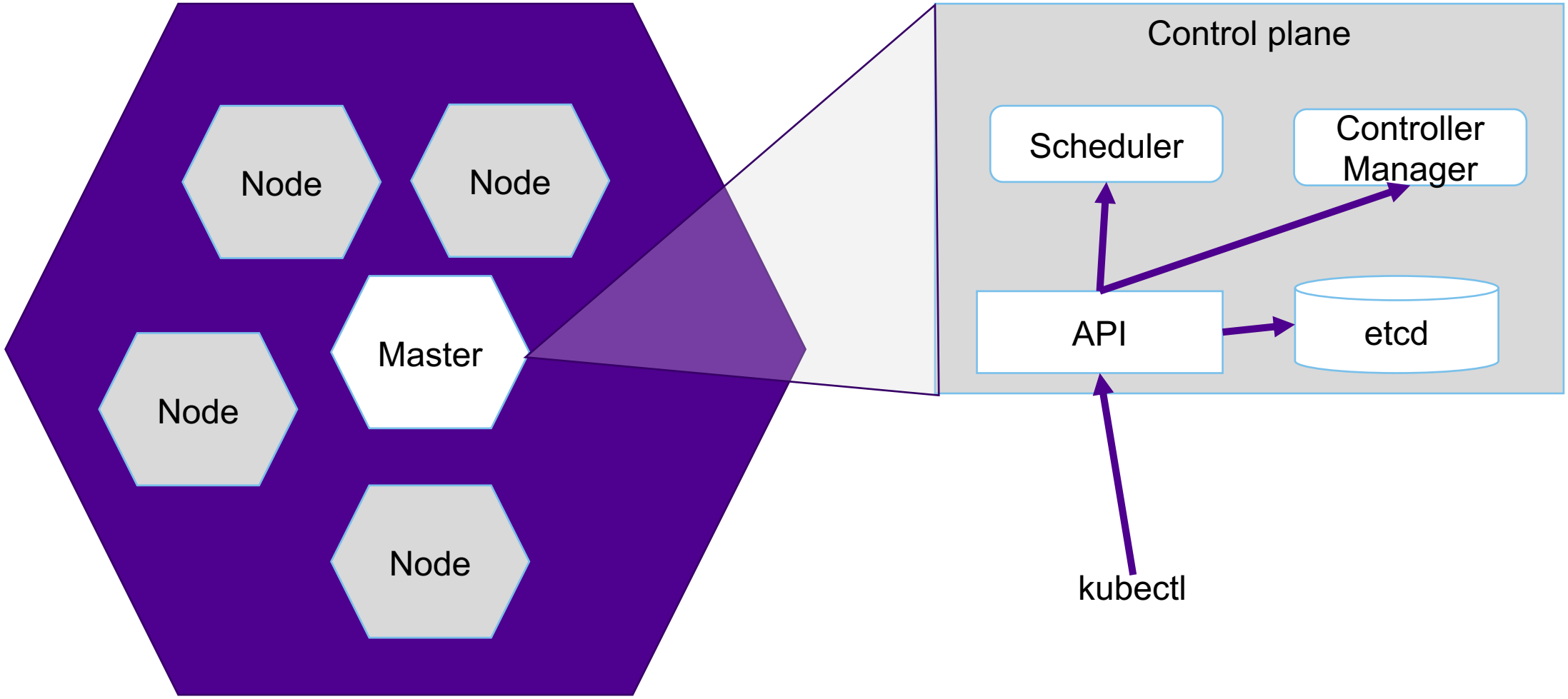


6. Update your app

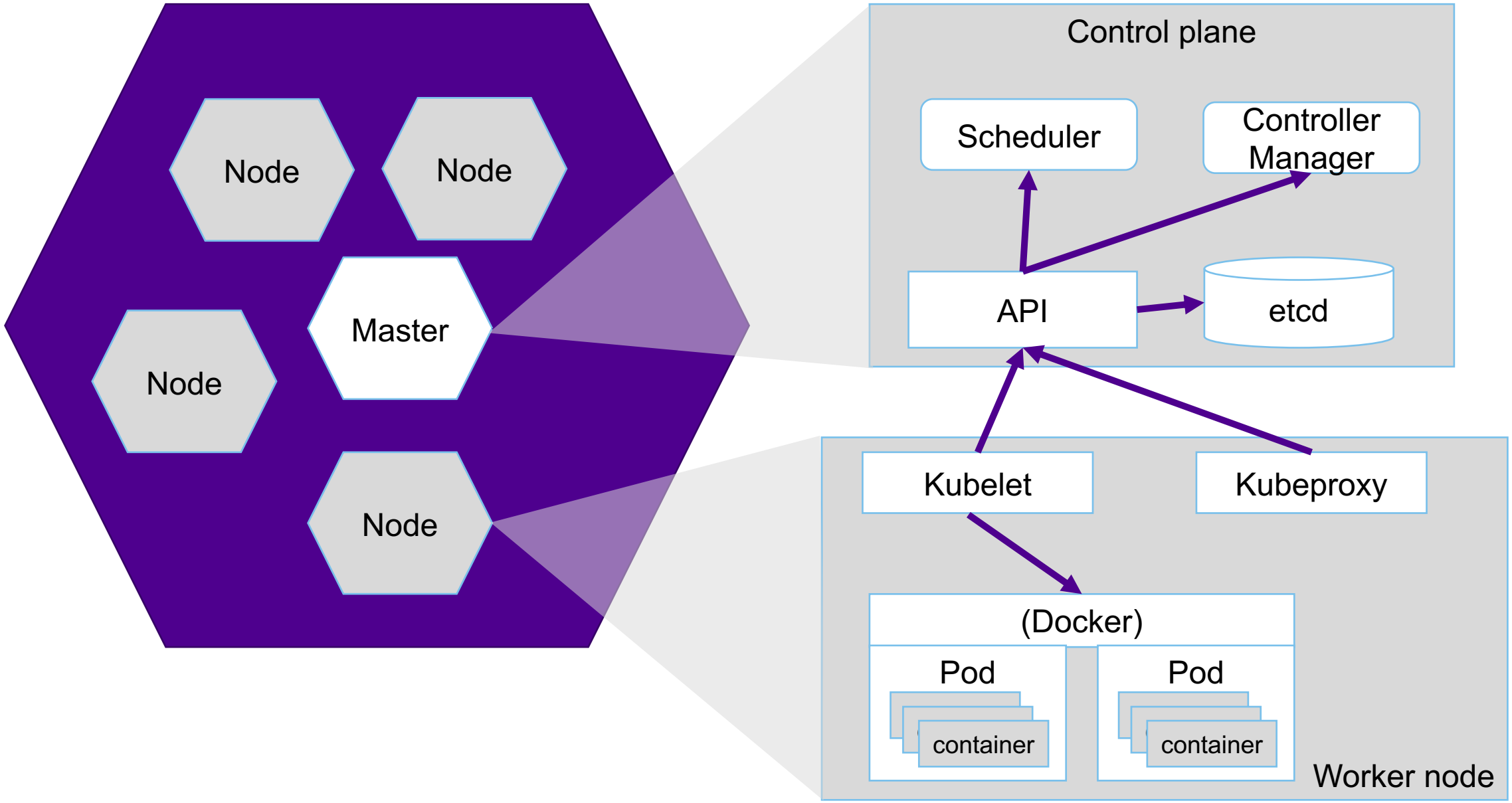
Kubernetes Kluster



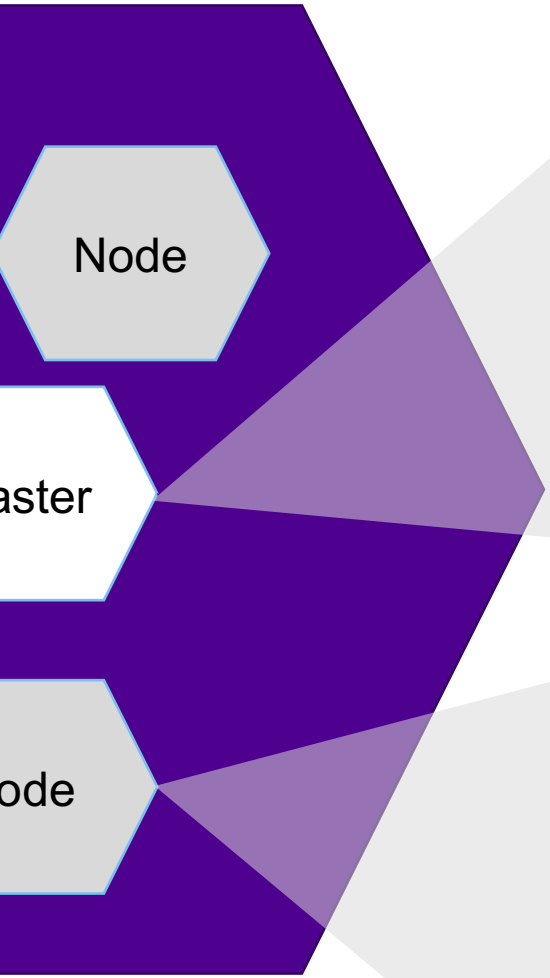
Kubernetes Kluster

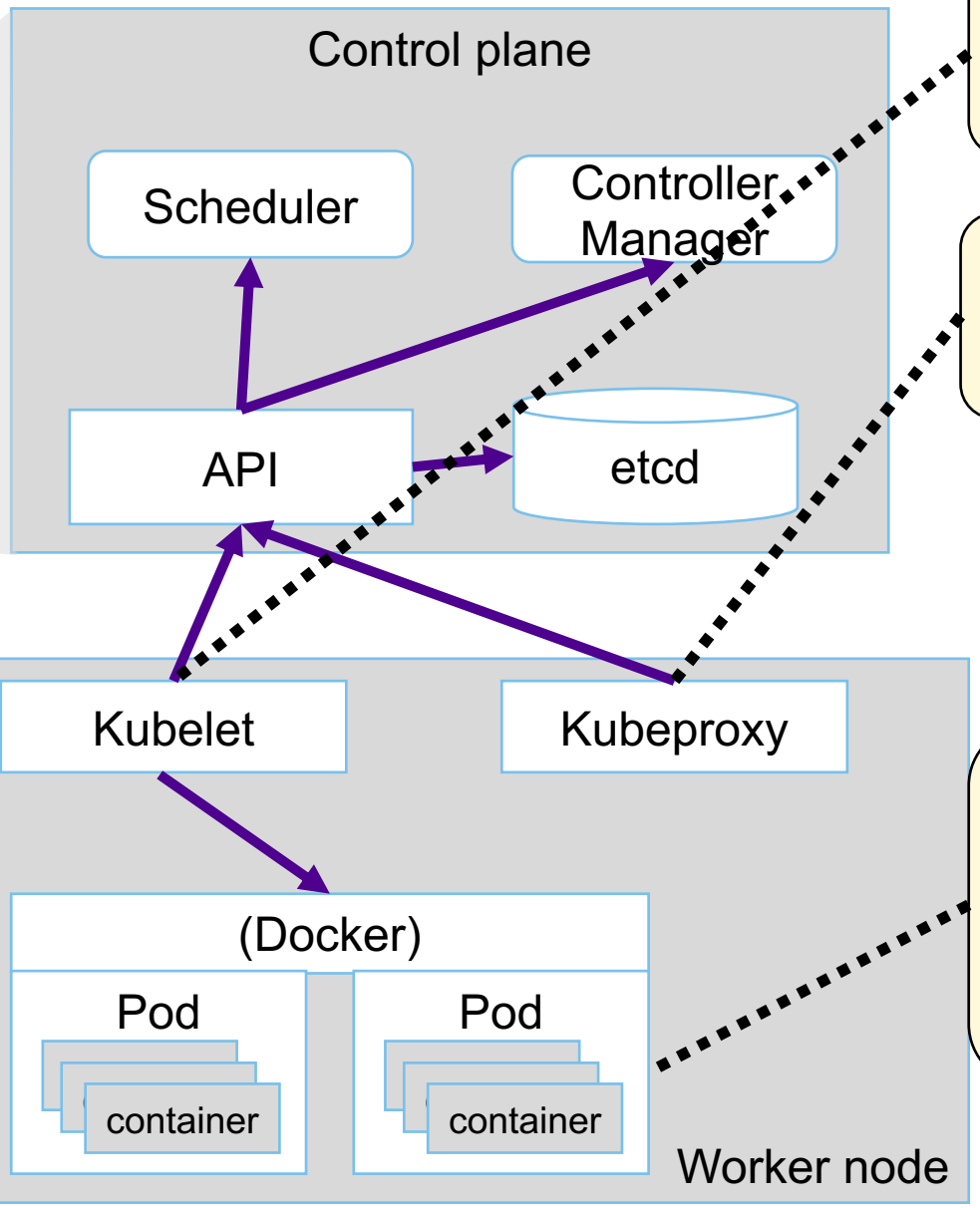


Kubernetes Kluster



Worker node



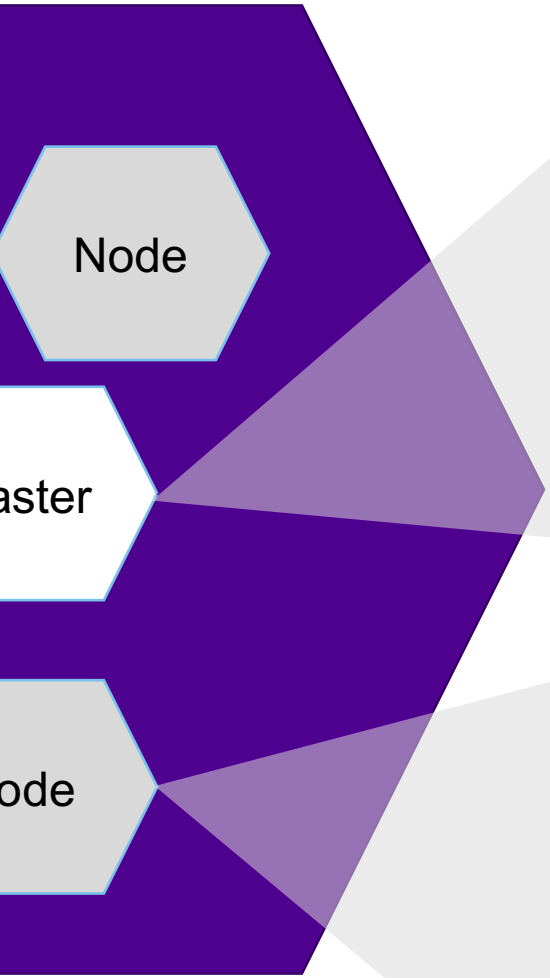


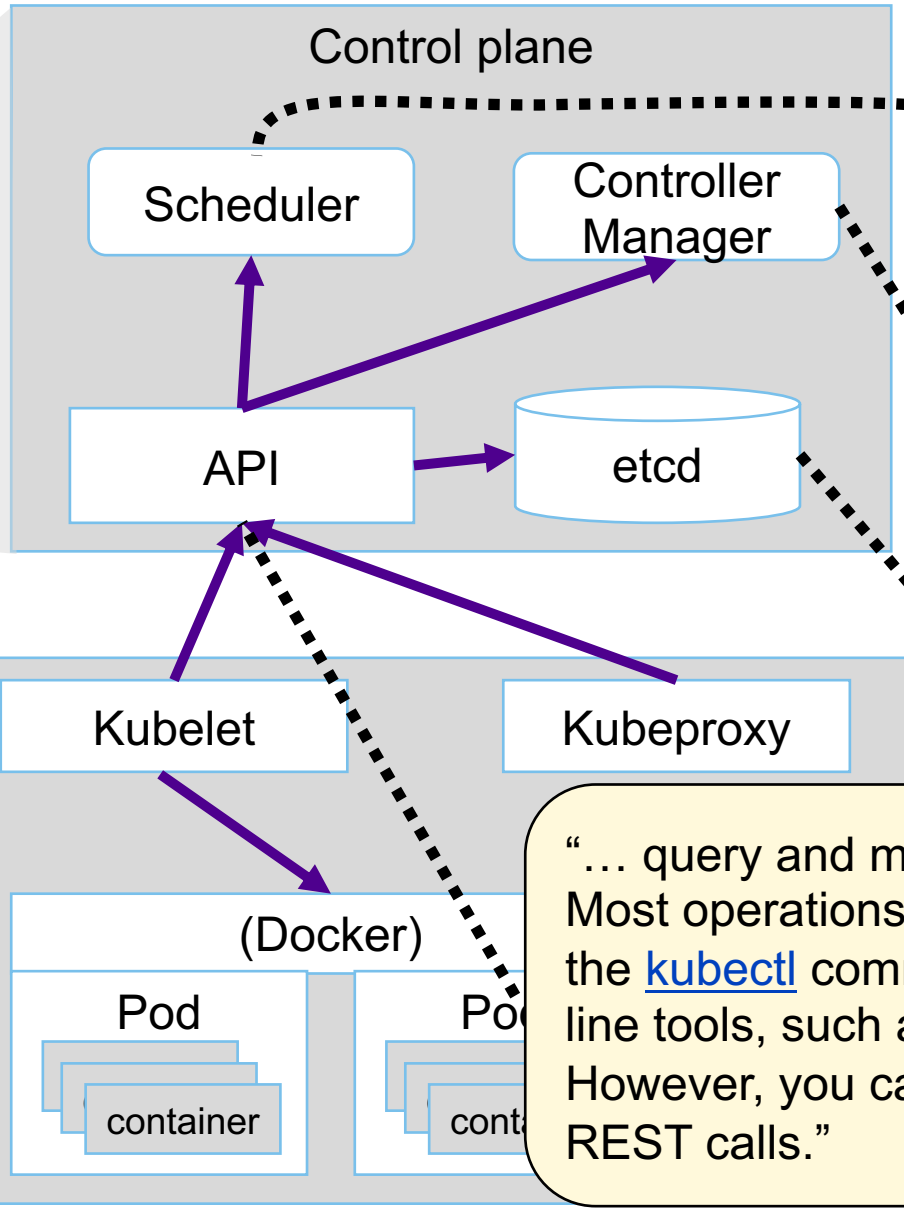
Agent to control the node. Starts (and stops) the containers.

Nework proxy

Smallest unit. Can have several containers that share resources with each other.
Every Pod has a unique IP address within the kluster

Control plane





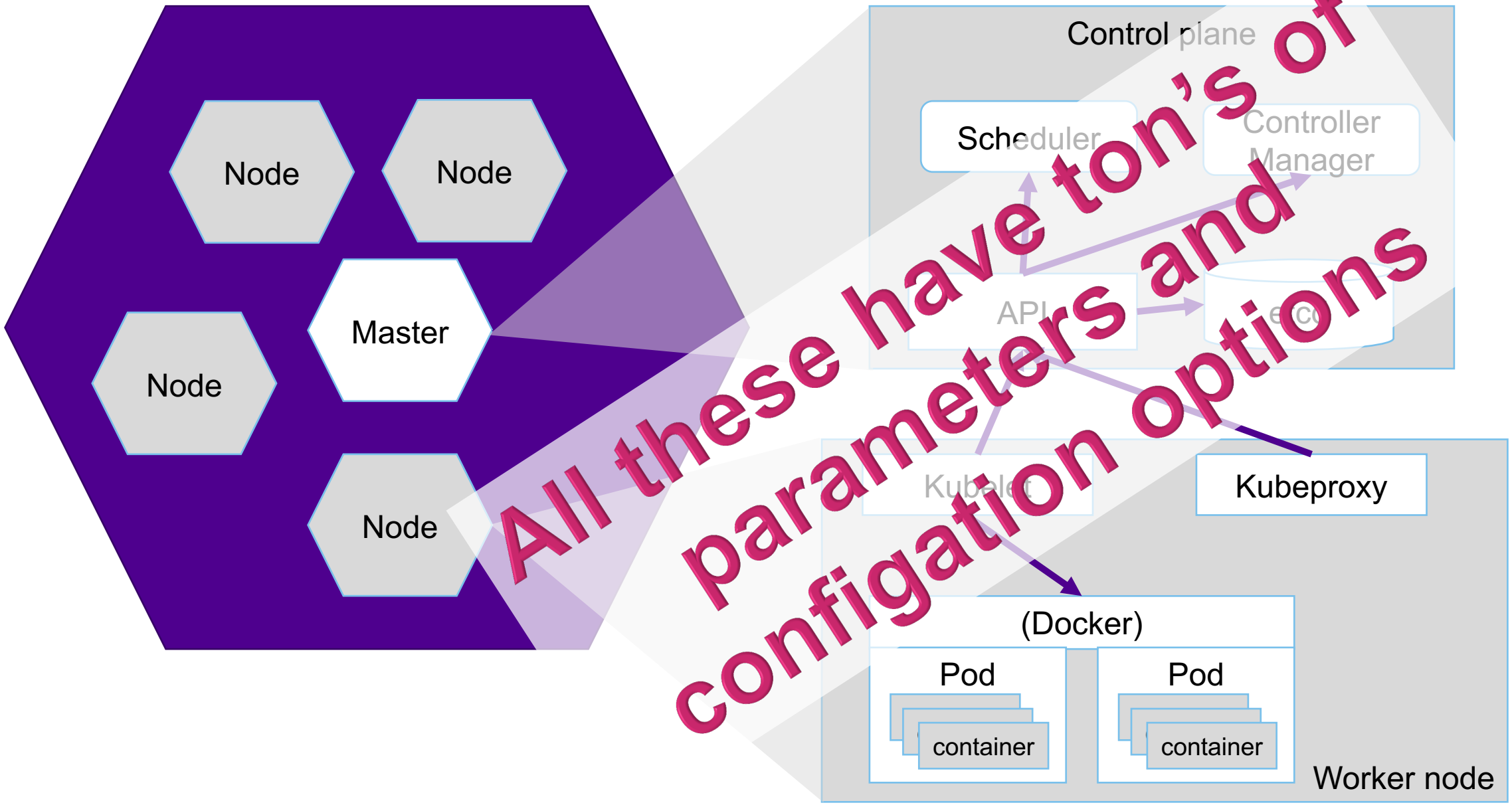
Sees that every Pod has a Node to run it.
Finds the most suitable node

" a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state"

Key-value store for all kind of configuration information

"… query and manipulate the state of objects … Most operations can be performed through the kubectl command-line interface or other command-line tools, such as kubeadm, which in turn use the API. However, you can also access the API directly using REST calls."

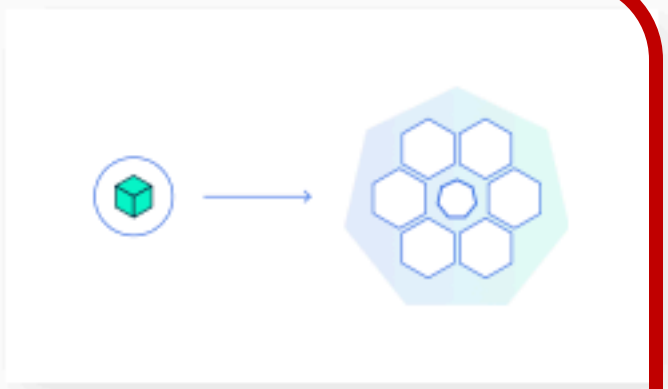
Kubernetes Kluster



Then next step

(source: https://kubernetes.io/docs/tutorials/kubernetes-basics/)

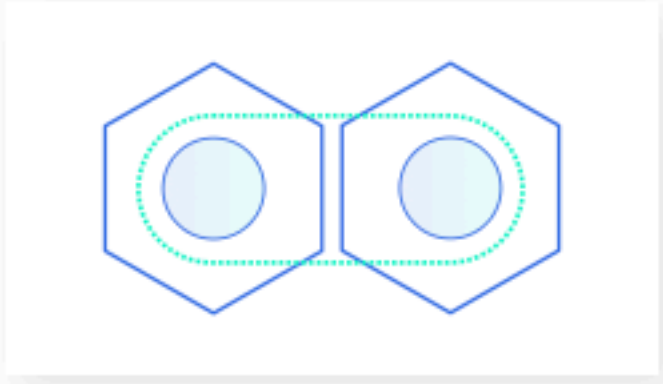
1. Create a Kubernetes cluster



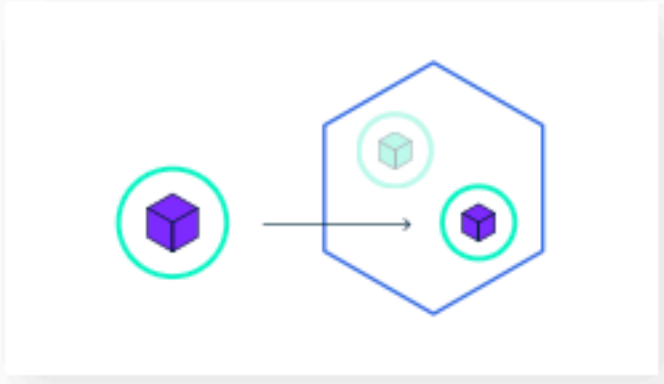
2. Deploy an app

3. Explore your app

4. Expose your app publicly



5. Scale up your app



6. Update your app

Deployment to Kubernetes

- Step 1: create Deployment (configuration)
- “A Deployment is responsible for creating and updating instances of your application”

```
$ kubectl apply -f
    https://k8s.io/examples/controllers/nginx-deployment.yaml
```

```
$ kubectl get deployments
```

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   0/3     0            0           1s
```

```
$ kubectl get deployments
```

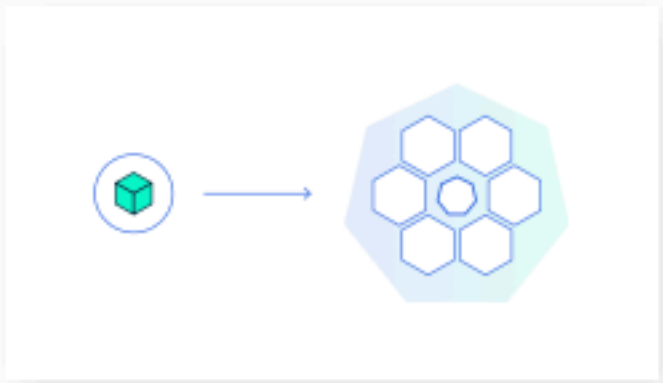
```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           18s
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

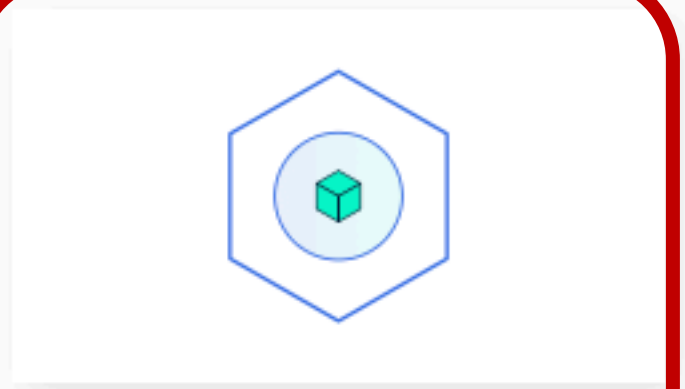
Then next step

(source: https://kubernetes.io/docs/tutorials/kubernetes-basics/)

1. Create a Kubernetes cluster

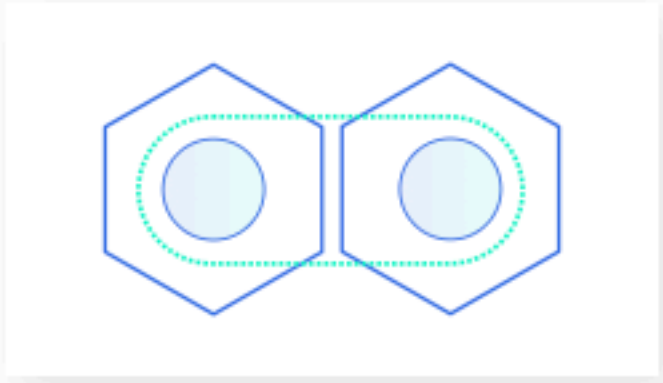


2. Deploy an app

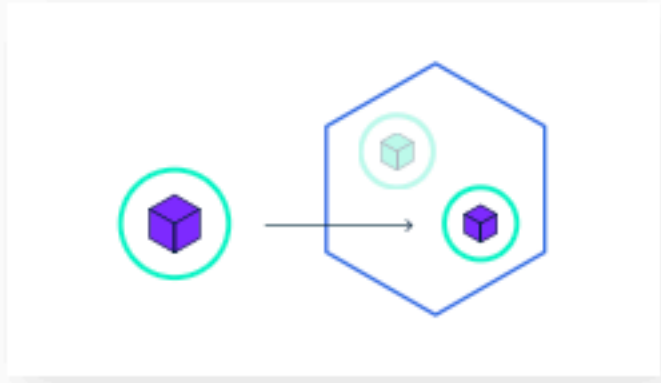


3. Explore your app

4. Expose your app publicly



5. Scale up your app



6. Update your app

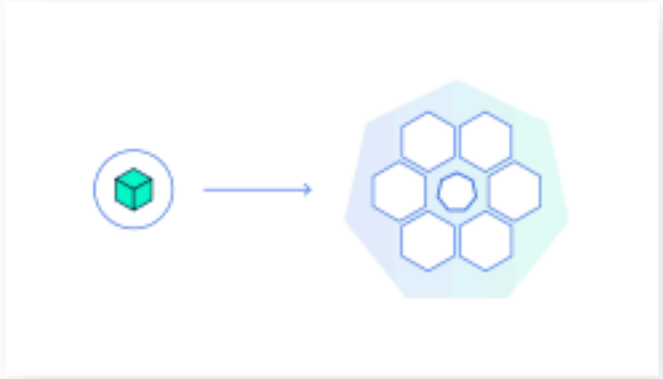
Examples of tools for trouble shooting

- **kubectl get** - list resources
- **kubectl describe** - show detailed information about a resource
- **kubectl logs** - print the logs from a container in a pod
- **kubectl exec** - execute a command on a container in a pod

Then next step

(source: https://kubernetes.io/docs/tutorials/kubernetes-basics/)

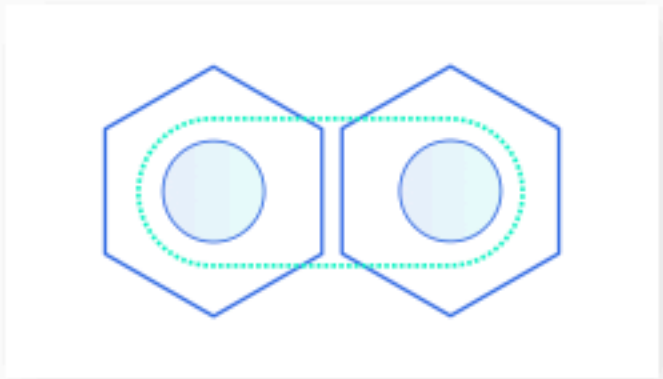
1. Create a Kubernetes cluster



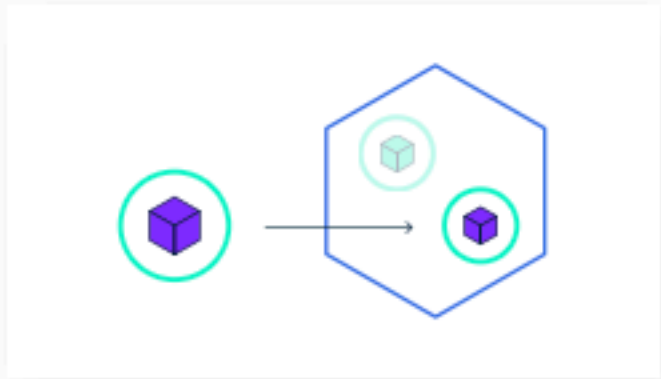
2. Deploy an app

3. Explore your app

4. Expose your app publicly



5. Scale up your app



6. Update your app

Tampere University

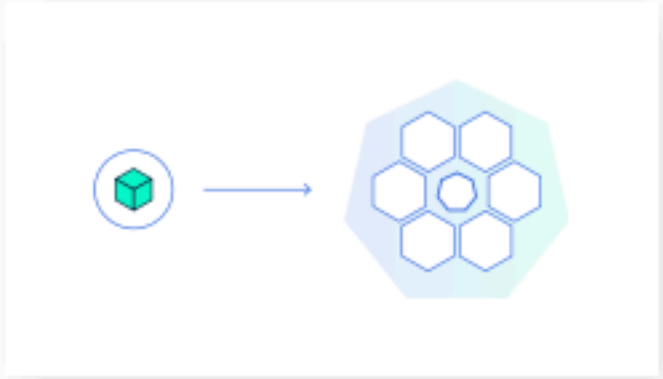
Kubernetes services

- Service
  - External IP and port
  - Load balancer
  - External name

Then next step

(source: https://kubernetes.io/docs/tutorials/kubernetes-basics/)

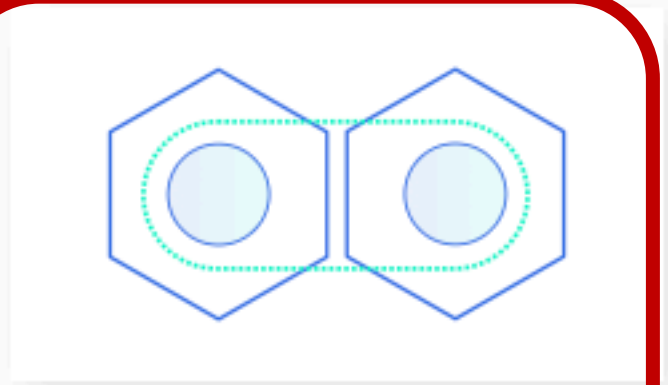
1. Create a Kubernetes cluster



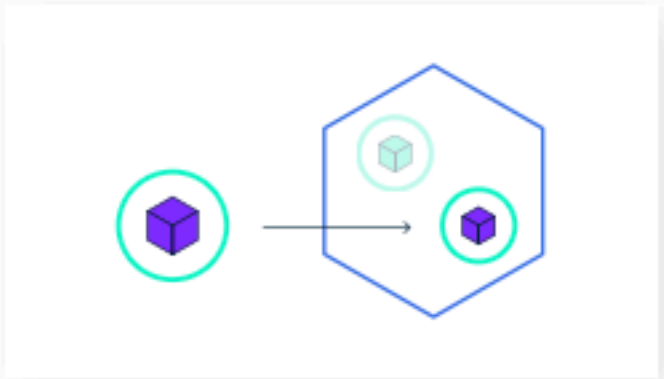
2. Deploy an app

3. Explore your app

4. Expose your app publicly



5. Scale up your app



6. Update your app

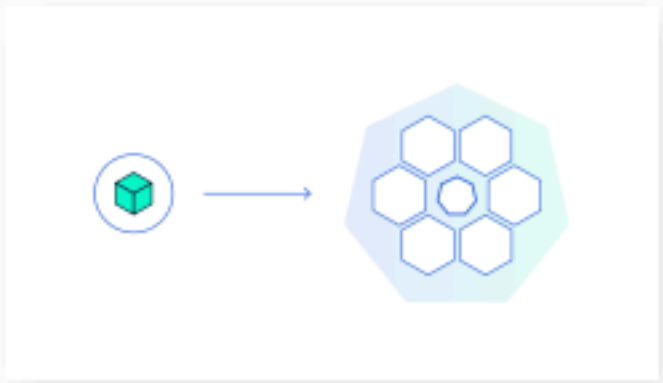
Scaling in Kubernets

- Scaling out a of Deployment increases the number of Pods to the new desired state.
- Kubernetes also supports auto scaling.
- Scaling to zero is also possible, and it will terminate all Pods of the specified Deployment.
- Services have an integrated load-balancer that will distribute network traffic to all Pods of an exposed Deployment.
- Services will monitor continuously the running Pods using endpoints, to ensure the traffic is sent only to available Pods.

Then next step

(source: https://kubernetes.io/docs/tutorials/kubernetes-basics/)

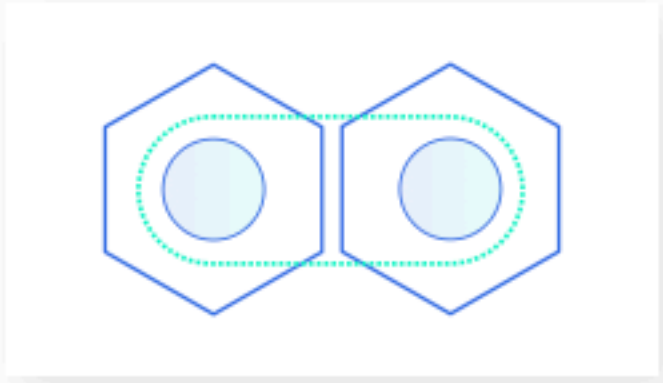
1. Create a Kubernetes cluster



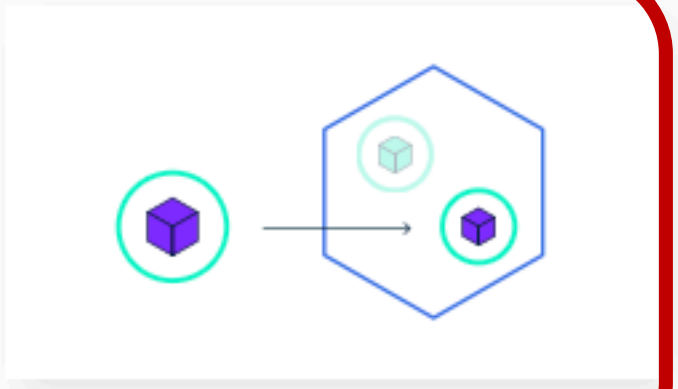
2. Deploy an app

3. Explore your app

4. Expose your app publicly



5. Scale up your app



6. Update your app

Updating an application

- Kubernetes uses **rolling updates**
- Zero down-time is the target

According to the marketing material supported features of Kubernetes include

- *Automatic binpacking*. Kubernetes automatically schedules the containers based on resource usage and constraints, without sacrificing the availability.
- *Self-healing*. Kubernetes automatically replaces and reschedules the containers from failed nodes. It also kills and restarts the containers which do not respond to health checks, based on existing rules/policy.
- *Horizontal scaling*. Kubernetes can automatically scale applications based on resource usage like CPU and memory. In some cases, it also supports dynamic scaling based on customer metrics.
- *Service discovery and Load balancing*. Kubernetes groups sets of containers and refers to them via a Domain Name System (DNS). This DNS is also called a Kubernetes service. Kubernetes can discover these services automatically, and load-balance requests between containers of a given service.
- *Automated rollouts and rollbacks*. Kubernetes can roll out and roll back new versions/configurations of an application, without introducing any downtime.
- *Secrets and configuration management*. Kubernetes can manage secrets and configuration details for an application without re-building the respective images. With secrets, we can share confidential information to our application without exposing it to the stack configuration, like on GitHub.
- *Storage orchestration*. With Kubernetes and its plugins, we can automatically mount local, external, and storage solutions to the containers in a seamless manner, based on software-defined storage (SDS).
- *Batch execution*. Besides long running jobs, Kubernetes also supports batch execution.

On how to configure containers or virtual machines

Alternative approaches

- Set-up everything when container is created
  - Very static
- Make the container to auto-update
  - You need to know in advance what might change
- ~~Put stuff to shared folder (use volume)~~
- Use configuration tools
  - Work also for full virtual machines and computers

Tampereen yliopisto
Tampere University

Ansible (https://www.ansible.com)

Automation engine for

- Provisioning
- Configuration Management
- App Deployment
- Continuous Delivery
- Security Automation
- Orchestration

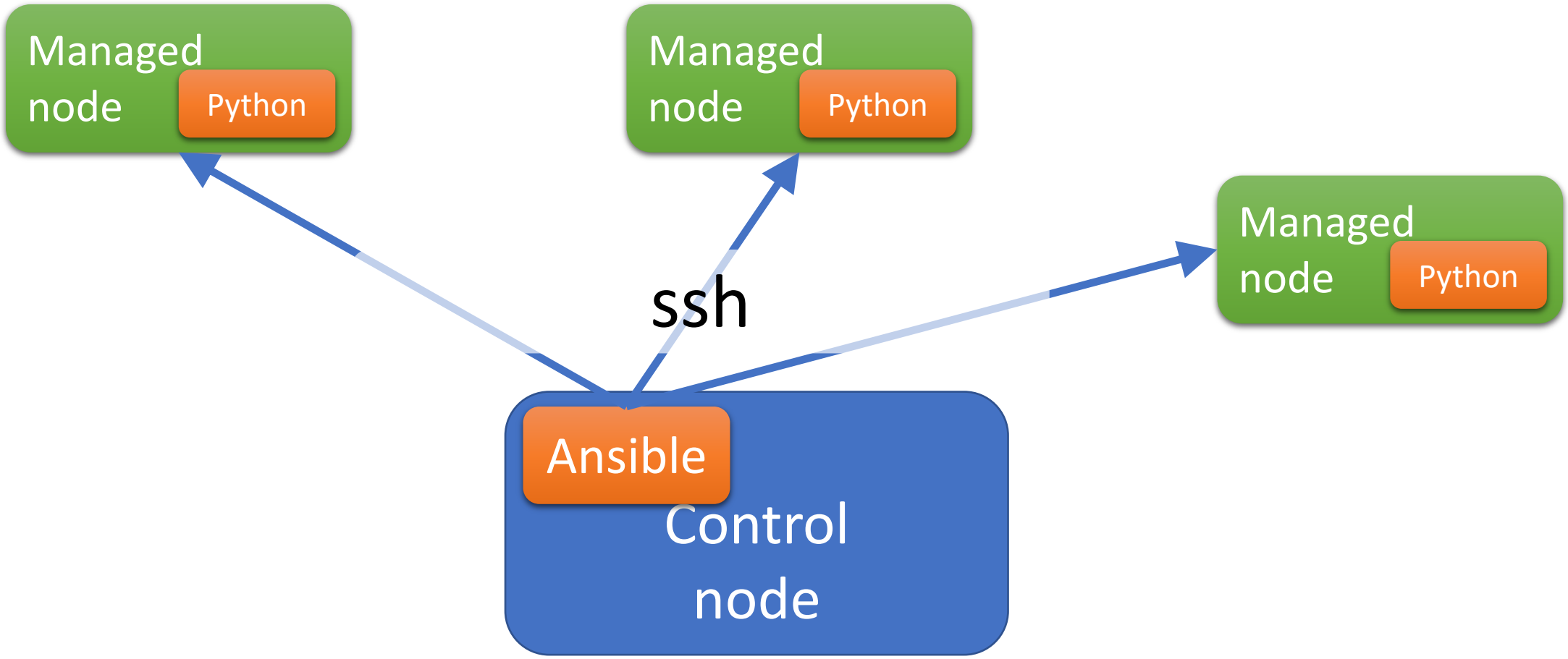
uses YAML, in the form of Ansible Playbooks

Tampereen yliopisto
Tampere University

Ansible

- Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default), and removes them when finished.
- Your library of modules can reside on any machine, and there are no servers, daemons, or databases required. Typically you'll work with your favorite terminal program, a text editor, and probably a version control system to keep track of changes to your content.
- A short video:
  - https://www.ansible.com/resources/videos/quick-start-video

Architecture

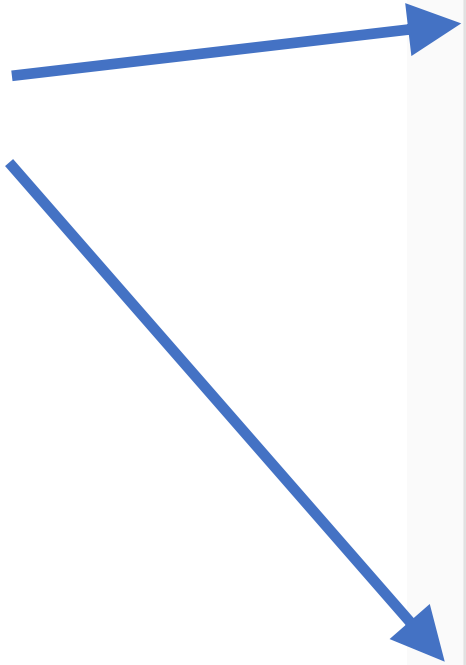


Example ansible playbook

```
---
- hosts: webservers
  vars:
     http_port: 80
     max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the
          latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config
          file.
    template:
      src: /srv/httpd.j2
      dest:/etc/httpd.conf
    notify:
      - restart apache
  - name: ensure apache is
          running
    service:
      name: httpd
      state: started

handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

There can be multiple plays



```yaml
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that postgresql is started
    service:
      name: postgresql
      state: started
```

Tampereen yliopisto
Tampere University

DevOps tools tend to fall into two categories:

- **Orchestration:** Deals with provisioning servers and other infrastructure including databases across clusters while handing over responsibility for managing the software running on the instances to configuration management tools.
- **Configuration management:** Focus on managing the software on infrastructure nodes, including installation and upgrades on servers already in existence

Tampereen yliopisto
Tampere University

Other alternatives

- **Chef:** Chef is written in the Ruby programming language and its CLI uses a Ruby-based DSL. Chef assumes an agent in the controlled nodes.
- **SaltStack:** Written in Python, SaltStack(Salt) holds your inventory's state on a master server, with YAML being the default format for storing configurations. SaltStack templates use the Jinja templating language, which will be familiar to Python users.
- **Puppet:** Another tool on the configuration management side, Puppet requires a master server, called the Puppet master, which stores the configuration of your infrastructure and pushes changes out to clients.
- **Terraform:** A different approach to both Ansible and Chef. Terraform focuses on setting up your entire infrastructure and provisioning servers. As such, it falls on the orchestration side of the spectrum, but it can be used alongside configuration focused tools like Chef and Ansible

Tampereen yliopisto
Tampere University

Terraform

- Declarative approach, too.
  - User writes a specification of the target state
- Pluggable architecture
  - Can have providers for AWS, Azure, gCloud
- Code – (plan –)apply

example.tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 2.70"
    }
  }
}
provider "aws" {
  profile = "default"
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami = "ami-830c94e3"
  instance_type = "t2.micro"
}
```

```
$ terraform show
# aws_instance.example:
resource "aws_instance" "example" {
    ami                          = "ami-830c94e3"
    arn                          = "arn:aws:ec2:us-east-1:130490850807:instance/i-0bbf06244e44211d1"
    associate_public_ip_address  = true
    availability_zone            = "us-west-2"
    cpu_core_count               = 1
    cpu_threads_per_core         = 1
    disable_api_termination      = false
    ebs_optimized                = false
    get_password_data            = false
    id                           = "i-0bbf06244e44211d1"
    instance_state               = "running"
    instance_type                = "t2.micro"
    ipv6_address_count           = 0
    ipv6_addresses               = []
    monitoring                   = false
    primary_network_interface_id = "eni-0f1ce5bdae258b015"
    private_dns                  = "ip-172-31-61-141.ec2.internal"
    private_ip                   = "172.31.61.141"
    public_dns                   = "ec2-54-166-19-naws.com"
    public_ip                    = "54.166.19.244"
    security_groups              = [
        "default",
    ]
    source_dest_check            = true
    subnet_id                    = "subnet-1facdf35"
    tenancy                      = "default"
    volume_tags                  = {}
    vpc_security_group_ids       = [
        "sg-5255f429",
    ]
```

BLE OF DEVOPS TOOLS (V3)

EMBED [

- Source Control Mgmt.
- Database Automation
- Continuous Integration
- Testing
- Configuration
- Deployment
- Containers
- Release Orchestration
- Cloud
- AIOps
- Analy
- Monit
- Secur
- Collal

23 Os **Fn** FitNesse	24 Fr **Ju** JUnit	25 Fr **Ka** Karma	26 Os **Su** SoapUI	27 En **Ch** Chef	28 Fr **Tf** Terraform	29 X Xe XL
41 Fr **Se** Selenium	42 Fr **Jm** JMeter	43 Os **Ja** Jasmine	44 Pd **Sl** Sauce Labs	45 Os **An** Ansible	46 Os **Ru** Rudder	47 O D
59 Os **Ga** Gatling	60 Fr **Tn** TestNG	61 Fm **Tt** Tricentis Tosca	62 Pd **Pe** Perfecto	63 En **Pu** Puppet	64 Os **Pa** Packer	65 A C
77 Fr **Cu** Cucumber	78 Os **Mc** Mocha	79 Os **Lo** Locust.io	80 En **Mf** Micro Focus UFT	81 Os **Sl** Salt	82 Os **Ce** CFEngine	83 E El

WikiPedia

Tool	Released by	Method	Approach	Written in	Comments
Pulumi	Pulumi[13]	Push	Declarative	Typescript, Python, Go	
Chef	Chef (2009)	Pull	Declarative and imperative	Ruby	
Otter	Inedo	Push	Declarative and imperative	-	Windows oriented
Puppet	Puppet (2005)	Pull	Declarative	C++ & Clojure from 4.0, Ruby	
SaltStack	SaltStack	Push and Pull	Declarative and imperative	Python	
CFEngine	CFEngine	Pull	Declarative	-	
Terraform	HashiCorp (2014)	Push	Declarative	Go	
DSC	Microsoft	Push/Pull	Declarative/Imperative	PowerShell	
Ansible / Ansible Tower	RedHat (2012)	Push	Declarative and imperative	Python	

Other tools include AWS CloudFormation, cdist, StackStorm and Juju (software).

Tampereen yliopisto
Tampere University

Finally

- Basic versions are often free/open source
- Commercial enterprise versions have nice graphical Web dashboard

Tampereen yliopisto
Tampere University

Automation scripts are programs
Infrastructure as code

- ”Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.”
- three approaches to IaC: declarative (functional) vs. imperative (procedural) vs. intelligent (environment aware)

```
tasks:
- name: ensure apache is at the
        latest version
  yum:
    name: httpd
    state: latest
- name: ensure that postgresql is started
    service:
      name: postgresql
      state: started
```

```
apt-get install …
```

From:

https://tutorials.cloudfoundry.org/cf-and-k8s/docs/comparing/

Design Approach

- **Cloud Foundry** is an **opinionated set of components** that are designed and distributed to work together.
- **Kubernetes** is a **flexible and extensible** system with a wide range of open source components from which you can choose, install, configure and maintain.

Built-in Functionality

- **Cloud Foundry** embodies an **opinionated workflow**, and automates the building of container images from application code, configuration of HTTPS access to your apps, and comes pre-configured with multi-tenancy access controls that are suitable for use in banks and governments.
- **Kubernetes** offers a considerable amount of flexibility, and **can be configured and extended to support virtually any workflow**. As such, you could assemble your own set of components on Kubernetes that replicate the functionality of Cloud Foundry.

Another: Heroku

https://devcenter.heroku.com/articles/how-heroku-works

- “Heroku lets you deploy, run and manage applications written in Ruby, Node.js, Java, Python, Clojure, Scala, Go and PHP.”
- “Heroku is a polyglot platform – it lets you build, run and scale applications in a similar manner across all the languages – utilizing the dependencies and Procfile. The Procfile exposes an architectural aspect of your application…”
- Architectural principles:
  - Strict separation of code and configuration, explicit dependency declaration, tight development iterations and parity between environments.
  - Applications are run as independent, lightweight, and stateless processes with quick startup and shutdown.
  - Execute auxiliary tasks in one-off processes and view application output via collated log-stream.

Automation challenges

- ”…provisioning scripts were considered error-prone and, according to developers, they did not work in some environments…”
- ”…automation of the network in was said to be difficult in addition to dealing with legacy system…”
- ”Networks are pretty hard. Some of the databases are pretty hard too because the old relational databases haven’t been designed to be clustered…”

Tampereen yliopisto
Tampere University

Reminders about infrastructure as code

All SW engineering principles should be applied.

- Testing
- Maintenance
- Documentation
- Version and configuration management

- Bugs may stop the whole engine

Tampereen yliopisto
Tampere University

Example of an alternative approach: CloudFoundry

- Claim: "*Cloud Foundry makes going from code to running apps as easy as a single cf push command. Don't spend your time writing infrastructure config for Kubernetes and Istio. Stay focused on your code.*"
- "First, equating the Cloud Foundry experience to a Kubernetes experience is like equating apples and walnuts. They're not at all related. Kubernetes is all about being an infrastructure abstraction, but that's not optimized for developers, and it has a more broadly applicable set of use cases – you can take a legacy app, slap it into a container and run it on Kubernetes; you could craft your own containers that are for a more modern architecture and run that on Kubernetes; and a whole bunch of things in between. The Cloud Foundry experience is focused on optimizing for the developer that's writing custom software for business or, in many cases, government applications. It's all about custom code."
- CloudFoundry may run on top of Kubernetes

Next week

- Recap and exam material