The Road To Declarative Infrastructure

From Basement Servers to GitOps

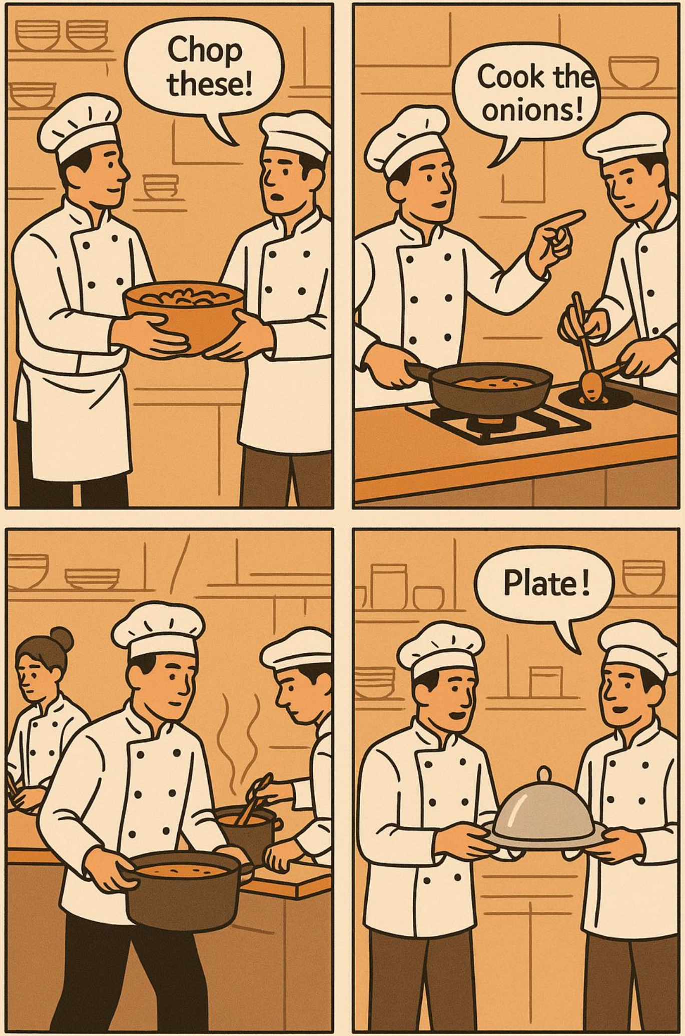
asana

Imperative infra - a kitchen story

You hold the artifact — the pot, the dough, the partially-prepared plate — and hand it off between steps.

You are the glue between chefs.

You manage timing, sequencing, context and the state of the order



asana

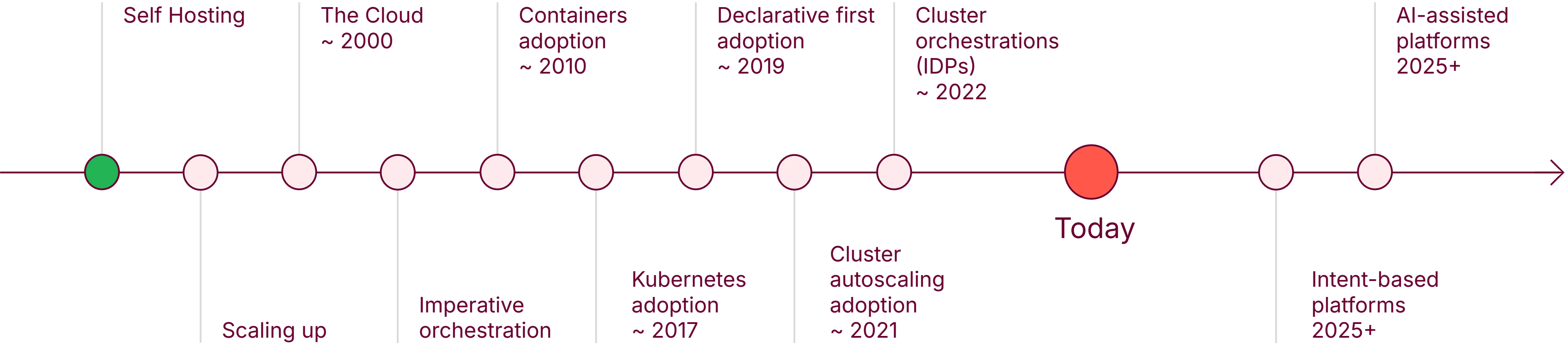
Declarative infra - a kitchen story

Each chef knows what their input should look like.

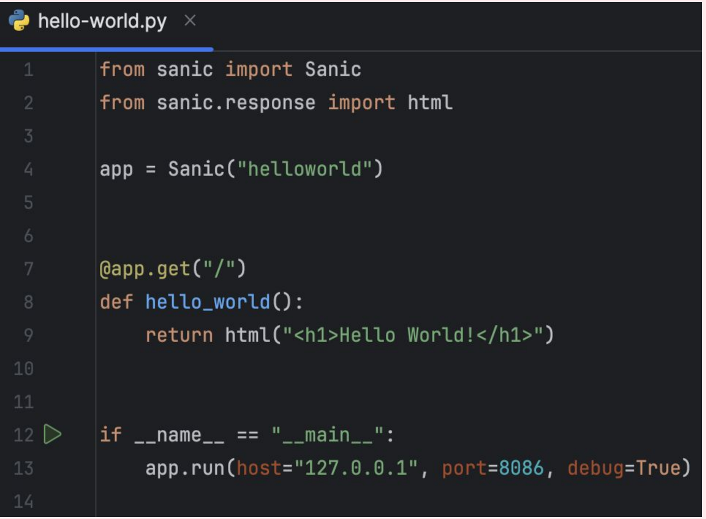
They pull the artifact when ready.

You define the outcome; the system coordinates ownership and flow.

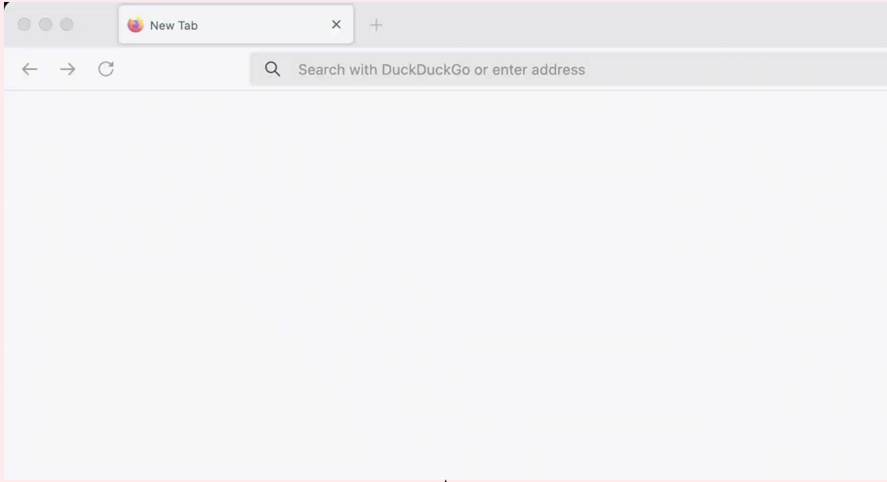
Timeline

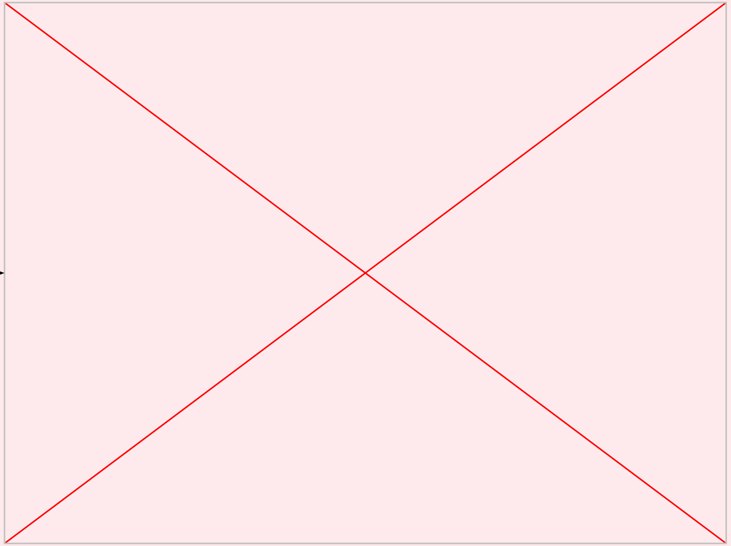

Self Hosting
Scaling up
The Cloud ~ 2000
Imperative orchestration
Containers adoption ~ 2010
Kubernetes adoption ~ 2017
Declarative first adoption ~ 2019
Cluster autoscaling adoption ~ 2021
Cluster orchestrations (IDPs) ~ 2022
Today
Intent-based platforms 2025+
AI-assisted platforms 2025+

Self Hosting - Localhost



```python
from sanic import Sanic
from sanic.response import html

app = Sanic("helloworld")


@app.get("/")
def hello_world():
    return html("<h1>Hello World!</h1>")


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=8086, debug=True)
```

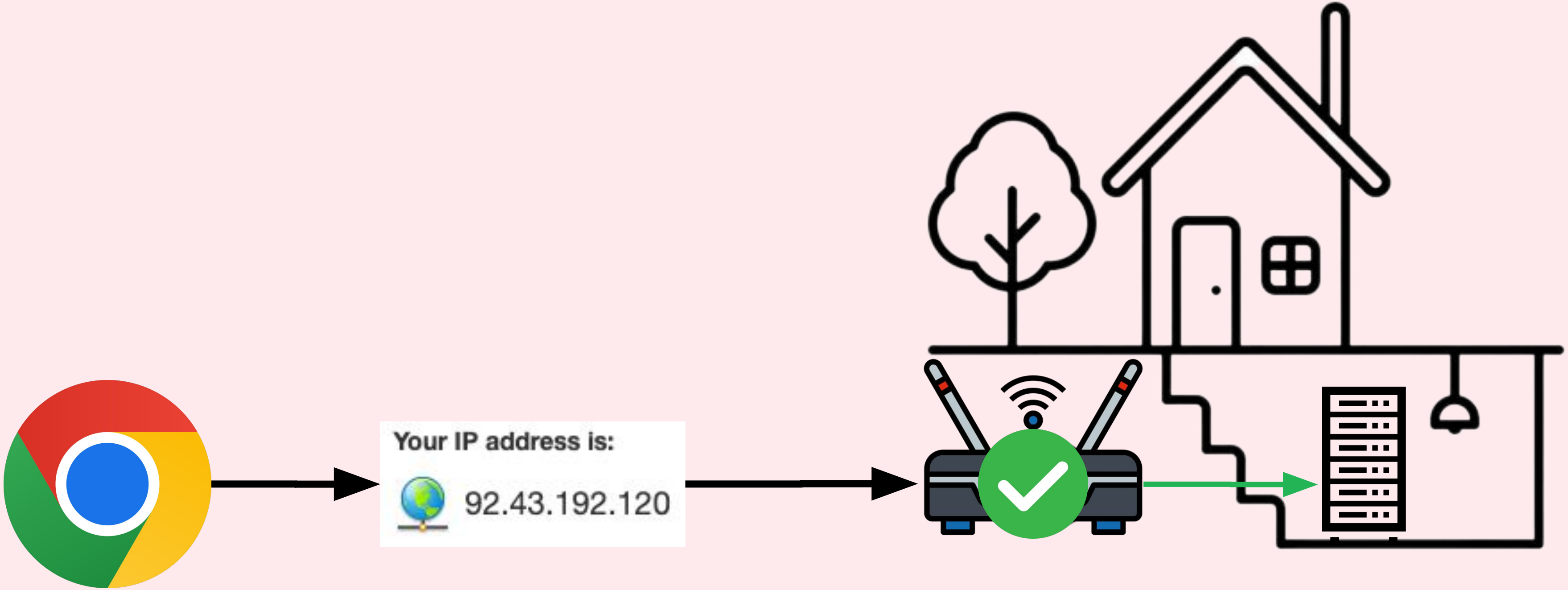




Server running on localhost:8086

Accessible on our on machine

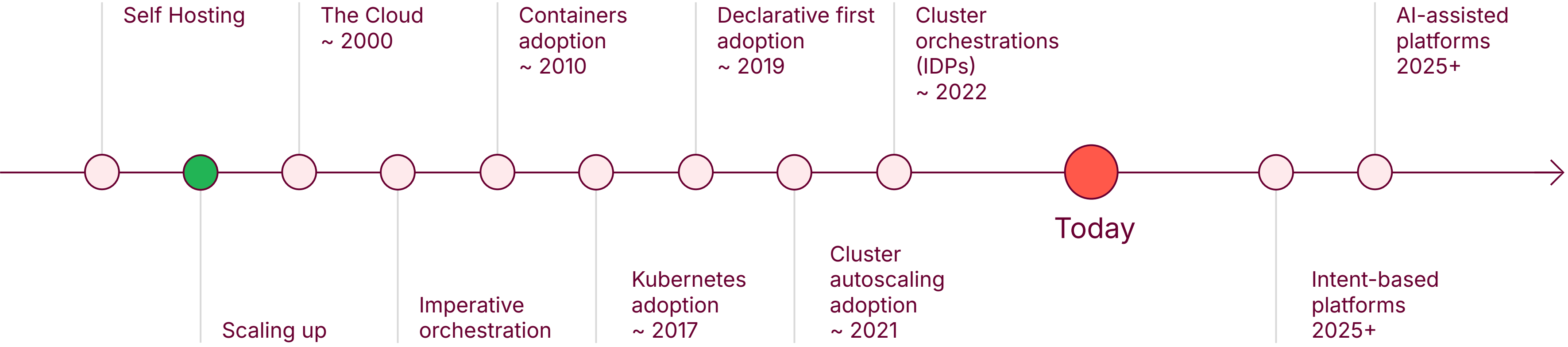
Self Hosting - Expose to the internet



Your IP address is:

92.43.192.120

Timeline



Self Hosting

Scaling up

The Cloud
~ 2000

Imperative
orchestration

Containers
adoption
~ 2010

Kubernetes
adoption
~ 2017

Declarative first
adoption
~ 2019

Cluster
autoscaling
adoption
~ 2021

Cluster
orchestrations
(IDPs)
~ 2022

Today

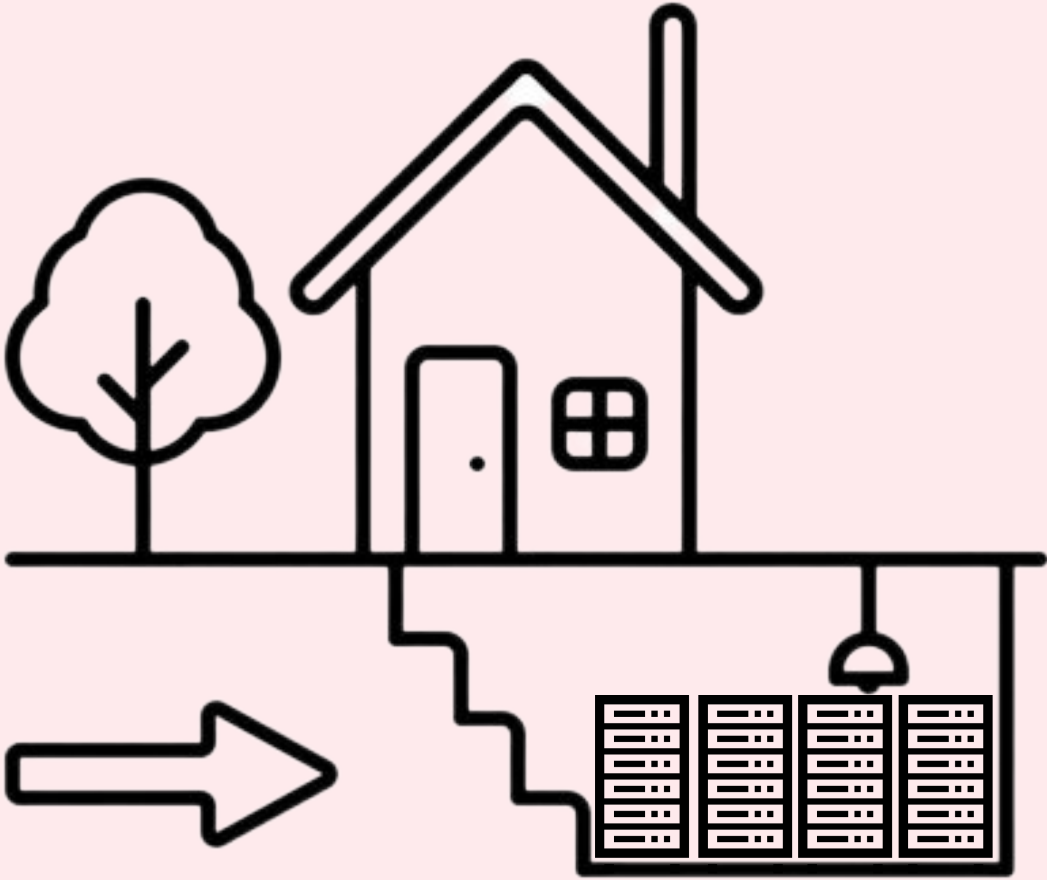
Intent-based
platforms
2025+

AI-assisted
platforms
2025+

Scaling up - more computers!

Problems

- Setting up the hardware
- Configuring the machine
- Run application server
- Maintaining
  - Hardware
  - Applications
  - Homogeneity / Drift

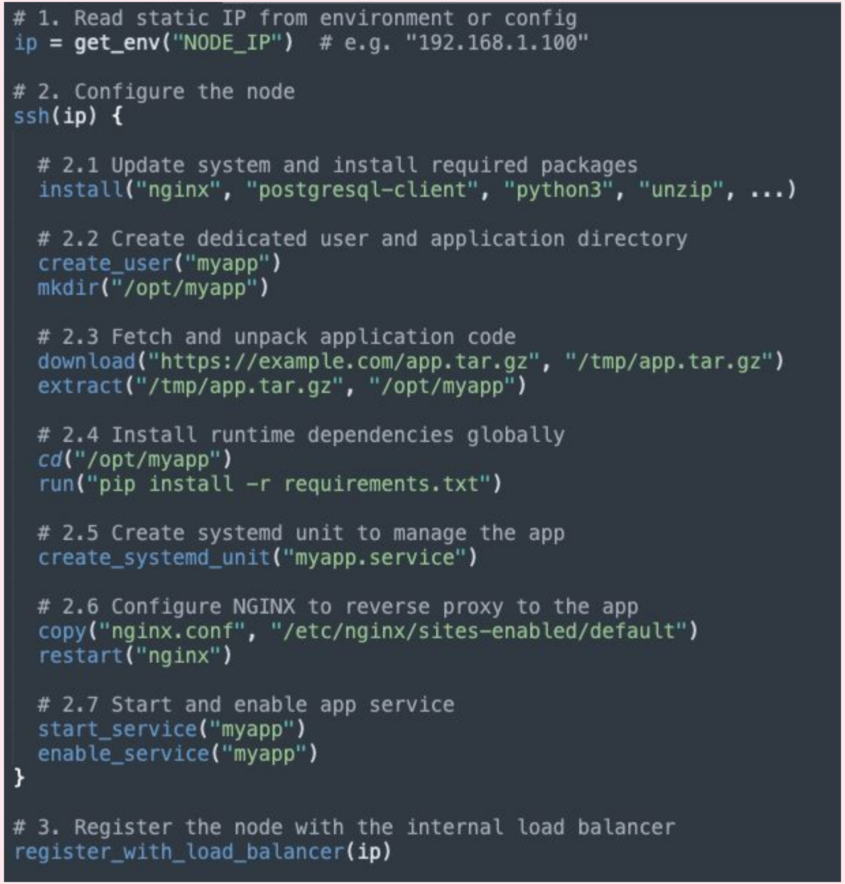


Scaling up - imperative scripting

Problems

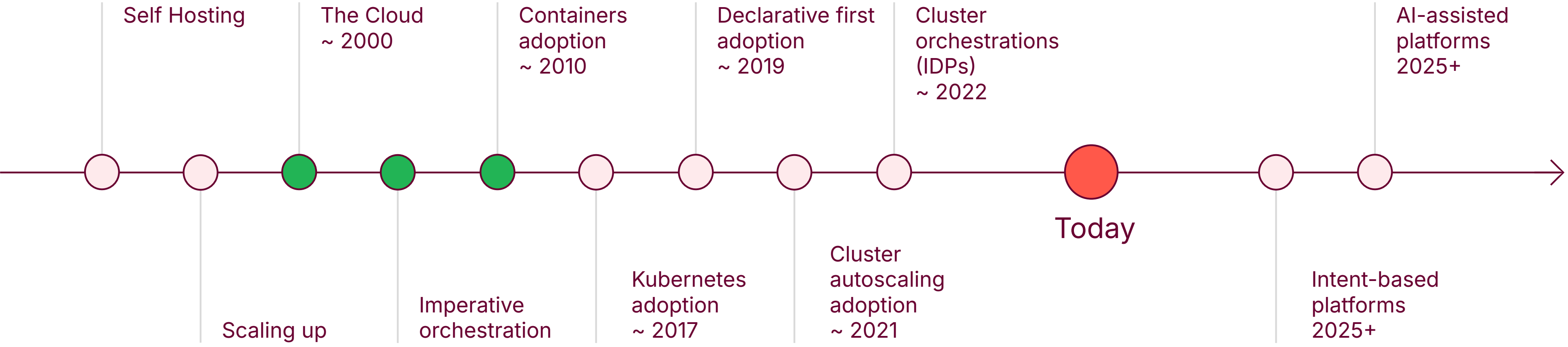
- Setting up the hardware
- Configuring the machine
- ~~Run application server~~
- Maintaining
    - Hardware
    - Applications
    - Homogeneity / Drift

Configuration via **imperative** scripting could go a long way...



```
# 1. Read static IP from environment or config
ip = get_env("NODE_IP")  # e.g. "192.168.1.100"

# 2. Configure the node
ssh(ip) {

  # 2.1 Update system and install required packages
  install("nginx", "postgresql-client", "python3", "unzip", ...)

  # 2.2 Create dedicated user and application directory
  create_user("myapp")
  mkdir("/opt/myapp")

  # 2.3 Fetch and unpack application code
  download("https://example.com/app.tar.gz", "/tmp/app.tar.gz")
  extract("/tmp/app.tar.gz", "/opt/myapp")

  # 2.4 Install runtime dependencies globally
  cd("/opt/myapp")
  run("pip install -r requirements.txt")

  # 2.5 Create systemd unit to manage the app
  create_systemd_unit("myapp.service")

  # 2.6 Configure NGINX to reverse proxy to the app
  copy("nginx.conf", "/etc/nginx/sites-enabled/default")
  restart("nginx")

  # 2.7 Start and enable app service
  start_service("myapp")
  enable_service("myapp")
}

# 3. Register the node with the internal load balancer
register_with_load_balancer(ip)
```

Timeline



Self Hosting

Scaling up

The Cloud
~ 2000

Imperative orchestration

Containers adoption
~ 2010

Kubernetes adoption
~ 2017

Declarative first adoption
~ 2019

Cluster autoscaling adoption
~ 2021

Cluster orchestrations (IDPs)
~ 2022

Today

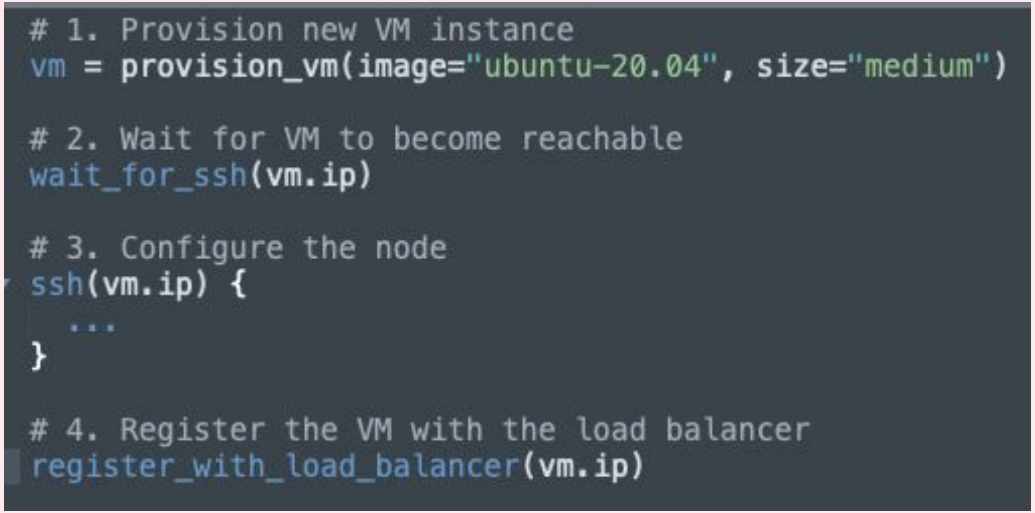
Intent-based platforms
2025+

AI-assisted platforms
2025+

The Cloud - Hardware via API calls

Problems

- ~~Setting up the hardware~~
- Configuring the machine
- ~~Run application server~~
- Maintaining
  - ~~Hardware~~
  - Applications
  - Homogeneity / Drift



```
# 1. Provision new VM instance
vm = provision_vm(image="ubuntu-20.04", size="medium")

# 2. Wait for VM to become reachable
wait_for_ssh(vm.ip)

# 3. Configure the node
ssh(vm.ip) {
    ...
}

# 4. Register the VM with the load balancer
register_with_load_balancer(vm.ip)
```

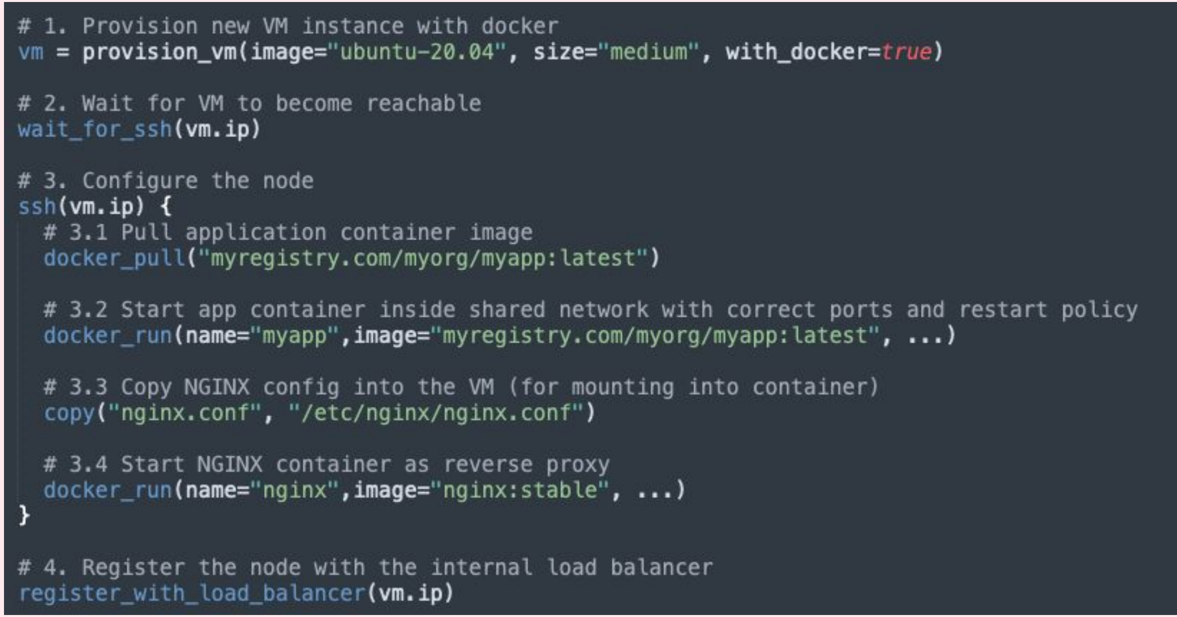
Containers - it works on all machines

Abstracted await the operating system

Standardised the interface between application and execution environment

Solved environment drift

Immutable, versioned artifacts

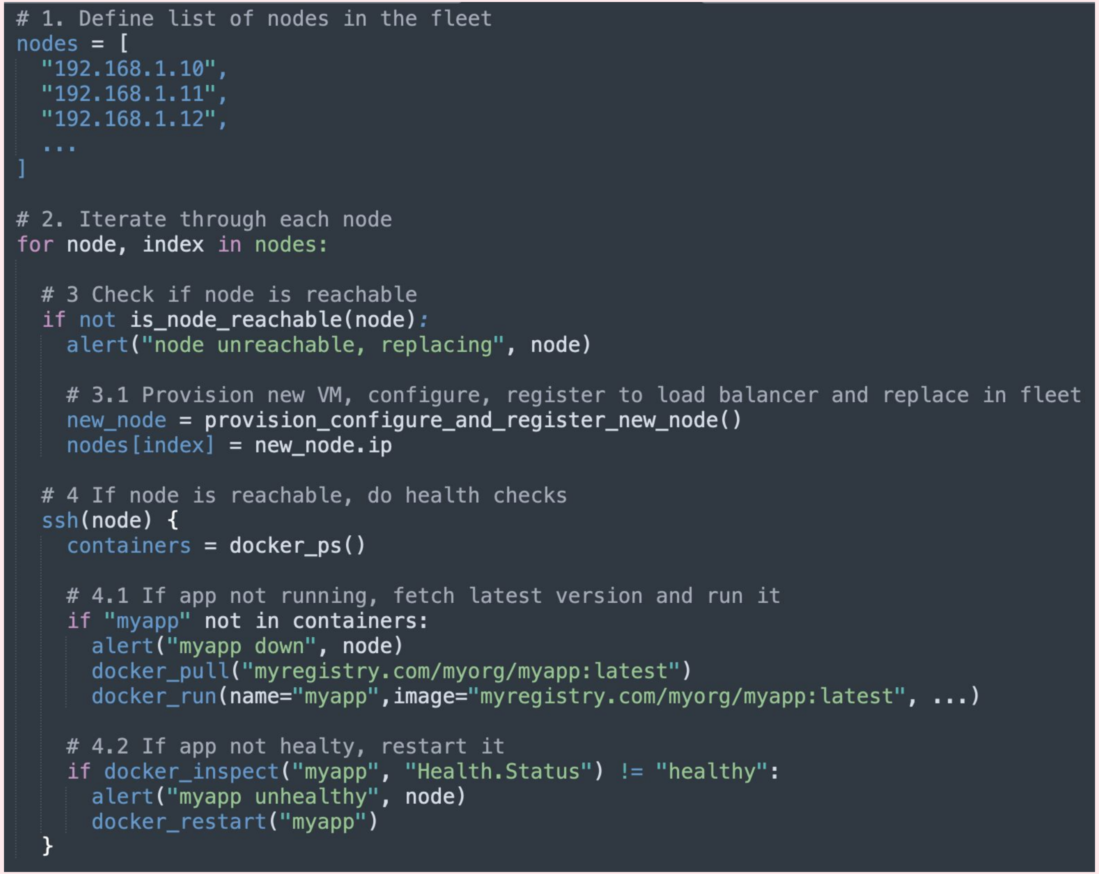


```
# 1. Provision new VM instance with docker
vm = provision_vm(image="ubuntu-20.04", size="medium", with_docker=true)

# 2. Wait for VM to become reachable
wait_for_ssh(vm.ip)

# 3. Configure the node
ssh(vm.ip) {
  # 3.1 Pull application container image
  docker_pull("myregistry.com/myorg/myapp:latest")

  # 3.2 Start app container inside shared network with correct ports and restart policy
  docker_run(name="myapp",image="myregistry.com/myorg/myapp:latest", ...)

  # 3.3 Copy NGINX config into the VM (for mounting into container)
  copy("nginx.conf", "/etc/nginx/nginx.conf")

  # 3.4 Start NGINX container as reverse proxy
  docker_run(name="nginx",image="nginx:stable", ...)
}

# 4. Register the node with the internal load balancer
register_with_load_balancer(vm.ip)
```

Imperative orchestration

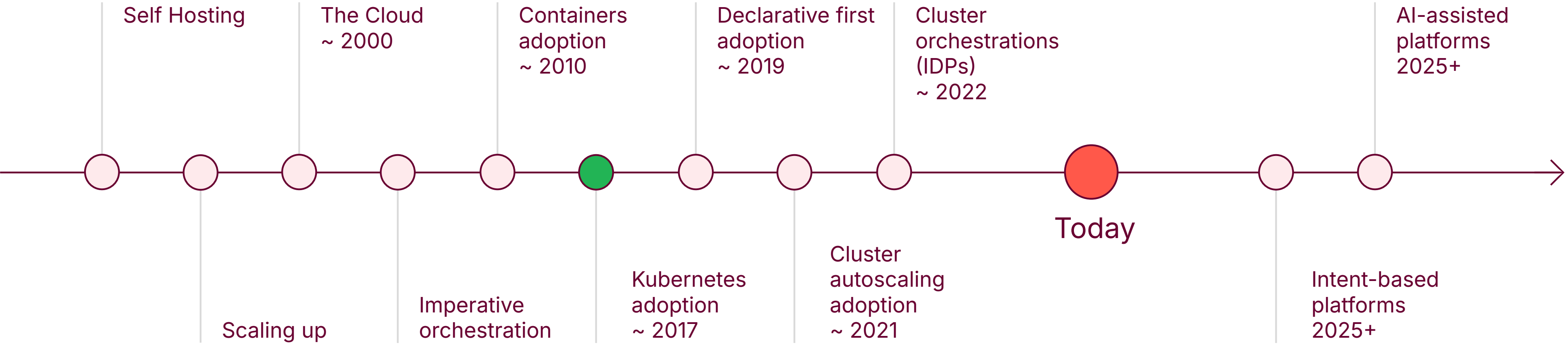
Problems

- ~~Setting up the hardware~~
- ~~Configuring the machine~~
- ~~Run application server~~
- Maintaining
  - ~~Hardware~~
  - ~~Applications~~
  - Homogeneity / Drift



```
# 1. Define list of nodes in the fleet
nodes = [
  "192.168.1.10",
  "192.168.1.11",
  "192.168.1.12",
  ...
]

# 2. Iterate through each node
for node, index in nodes:

  # 3 Check if node is reachable
  if not is_node_reachable(node):
    alert("node unreachable, replacing", node)

    # 3.1 Provision new VM, configure, register to load balancer and replace in fleet
    new_node = provision_configure_and_register_new_node()
    nodes[index] = new_node.ip

  # 4 If node is reachable, do health checks
  ssh(node) {
    containers = docker_ps()

    # 4.1 If app not running, fetch latest version and run it
    if "myapp" not in containers:
      alert("myapp down", node)
      docker_pull("myregistry.com/myorg/myapp:latest")
      docker_run(name="myapp",image="myregistry.com/myorg/myapp:latest", ...)

    # 4.2 If app not healty, restart it
    if docker_inspect("myapp", "Health.Status") != "healthy":
      alert("myapp unhealthy", node)
      docker_restart("myapp")
  }
```

Timeline



Self Hosting

Scaling up

The Cloud
~ 2000

Imperative
orchestration

Containers
adoption
~ 2010

Kubernetes
adoption
~ 2017

Declarative first
adoption
~ 2019

Cluster
autoscaling
adoption
~ 2021

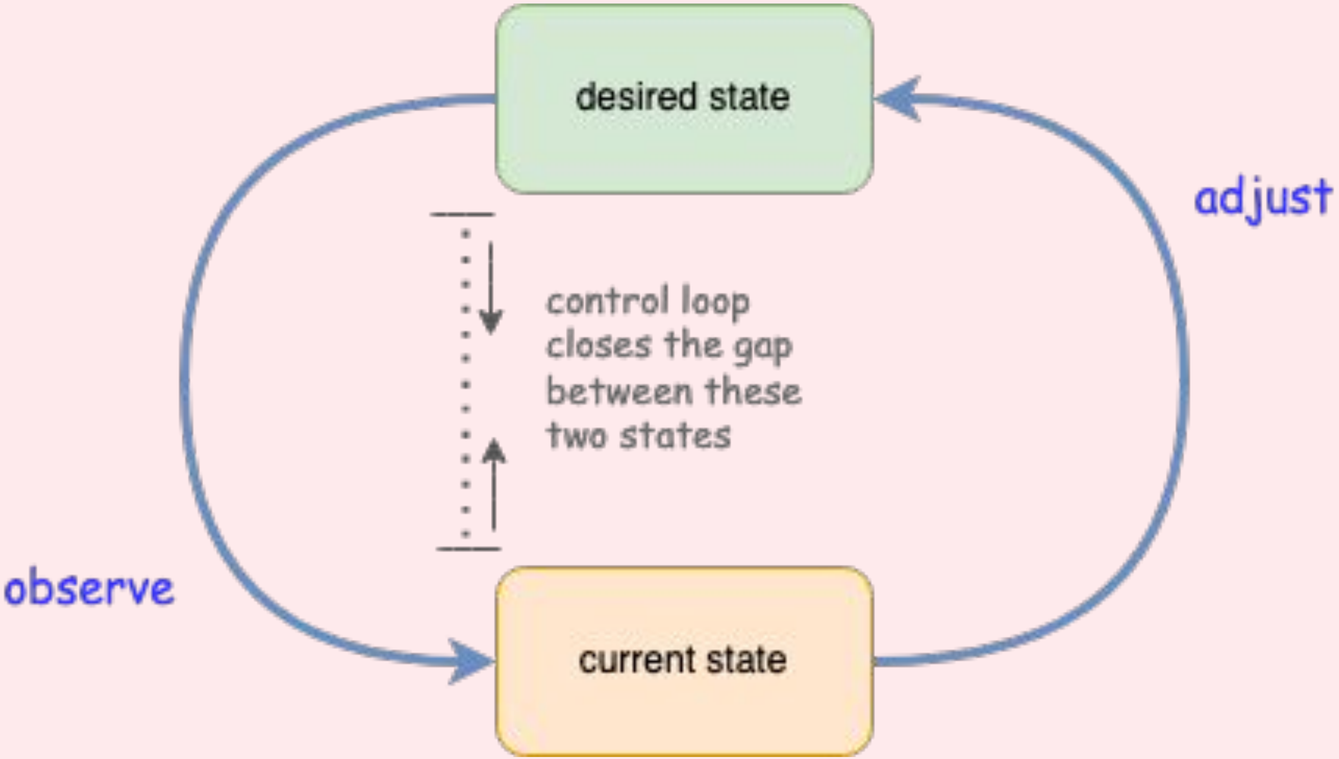
Cluster
orchestrations
(IDPs)
~ 2022

Today

Intent-based
platforms
2025+

AI-assisted
platforms
2025+

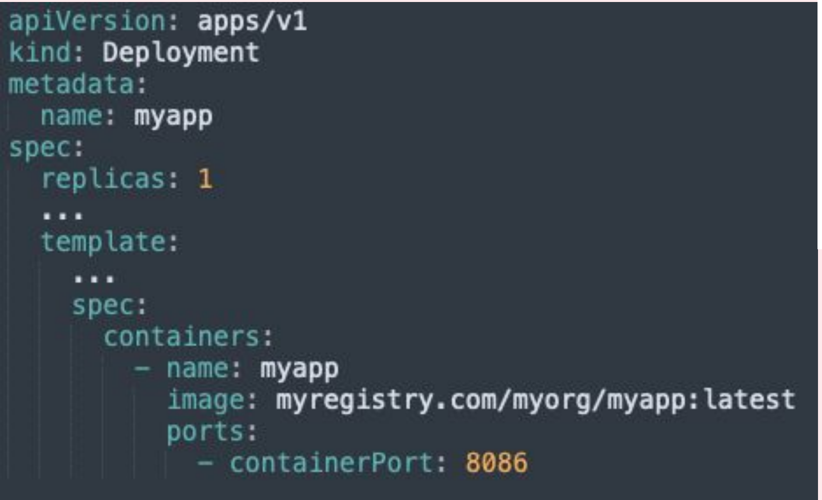
Kubernetes - Declarative interface for everything



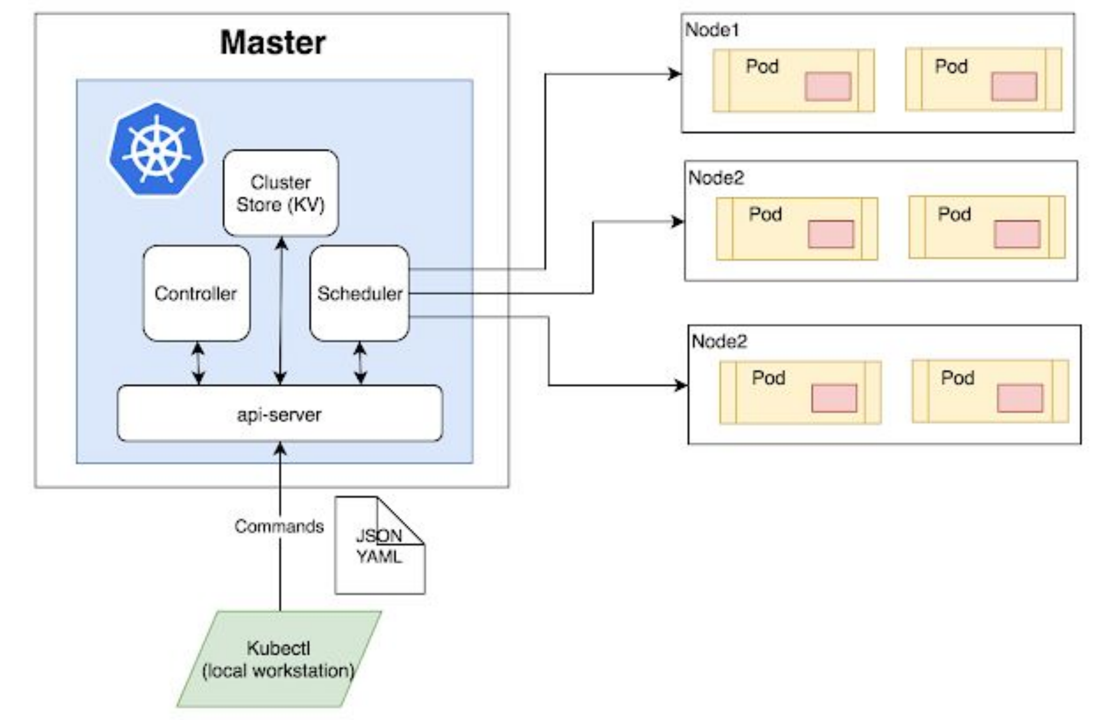
Kubernetes - Scheduling Declarative

Problems

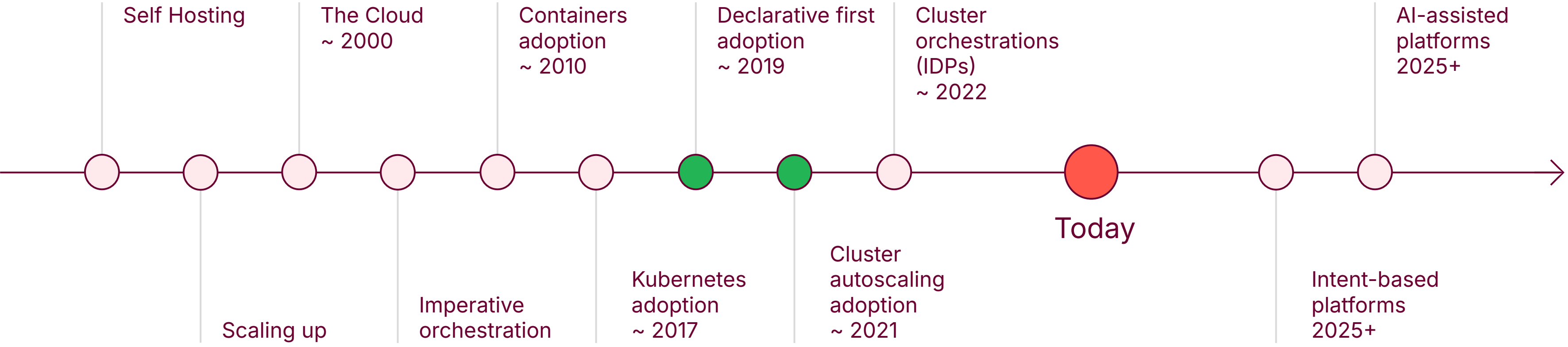
- ~~Setting up the hardware~~
- ~~Configuring the machine~~
- ~~Run application server~~
- ~~Maintaining~~
  - ~~Hardware~~
  - ~~Applications~~
  - ~~Homogeneity / Drift~~



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 1
  ...
  template:
    ...
    spec:
      containers:
        - name: myapp
          image: myregistry.com/myorg/myapp:latest
          ports:
            - containerPort: 8086
```



Timeline


Self Hosting
Scaling up
The Cloud ~ 2000
Imperative orchestration
Containers adoption ~ 2010
Kubernetes adoption ~ 2017
Declarative first adoption ~ 2019
Cluster autoscaling adoption ~ 2021
Cluster orchestrations (IDPs) ~ 2022
Today
Intent-based platforms 2025+
AI-assisted platforms 2025+

Declarative first adoption - GitOps

We’re continuously pushing declarative infrastructure up one abstraction layer

Declarative state stored in git

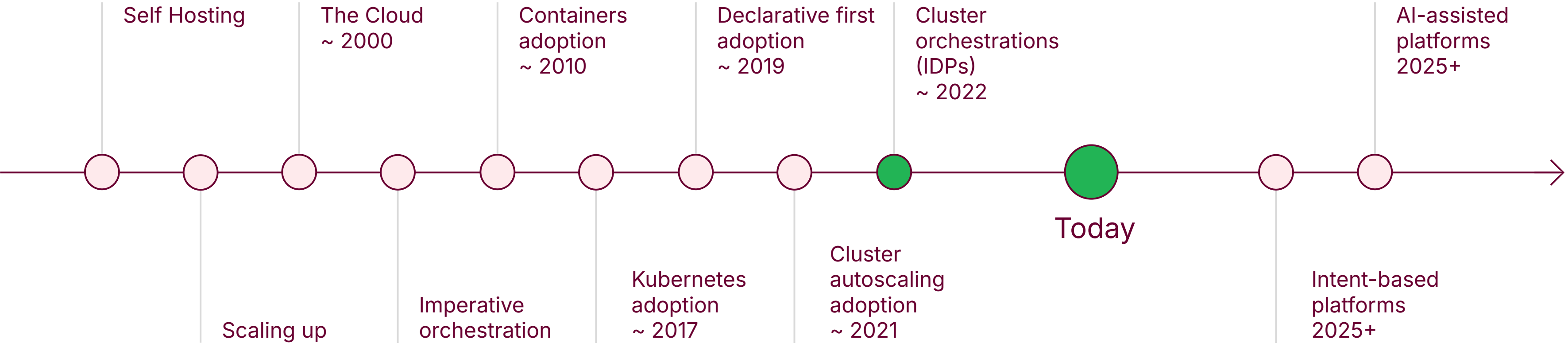
Moved from push to pull by controllers

Infrastructure versioned, just like your applications

Infrastructure rollbacks, just like your applications

Application and infrastructure versions rolled out with the same pipeline

Timeline



Self Hosting

Scaling up

The Cloud
~ 2000

Imperative
orchestration

Containers
adoption
~ 2010

Kubernetes
adoption
~ 2017

Declarative first
adoption
~ 2019

Cluster
autoscaling
adoption
~ 2021

Cluster
orchestrations
(IDPs)
~ 2022

Today

Intent-based
platforms
2025+

AI-assisted
platforms
2025+

Thanks everyone!