Nerdalize

Advanced Scheduling for heating showers

Ad van der Veer

Digital Product Engineer at Nerdalize

Talk Overview

- **Part 0:** Introducing Nerdalize
- **Part 1:** Our use case: scheduling heat
- **Part 2:** Creating a basic scheduler
- **Part 3:** Creating an advanced scheduler
- **Part 4:** Future Possibilities and Wrap-up

How computing was done...

How computing is done today...

The problem

Datacenters are incredibly cost, capital and energy **inefficient**

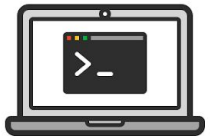
Our solution:

We place our cloud servers in homes instead of a datacenter

Going **datacenterless** creates a
unique value proposition with only winners

€200 / year
heat savings

for the home owner



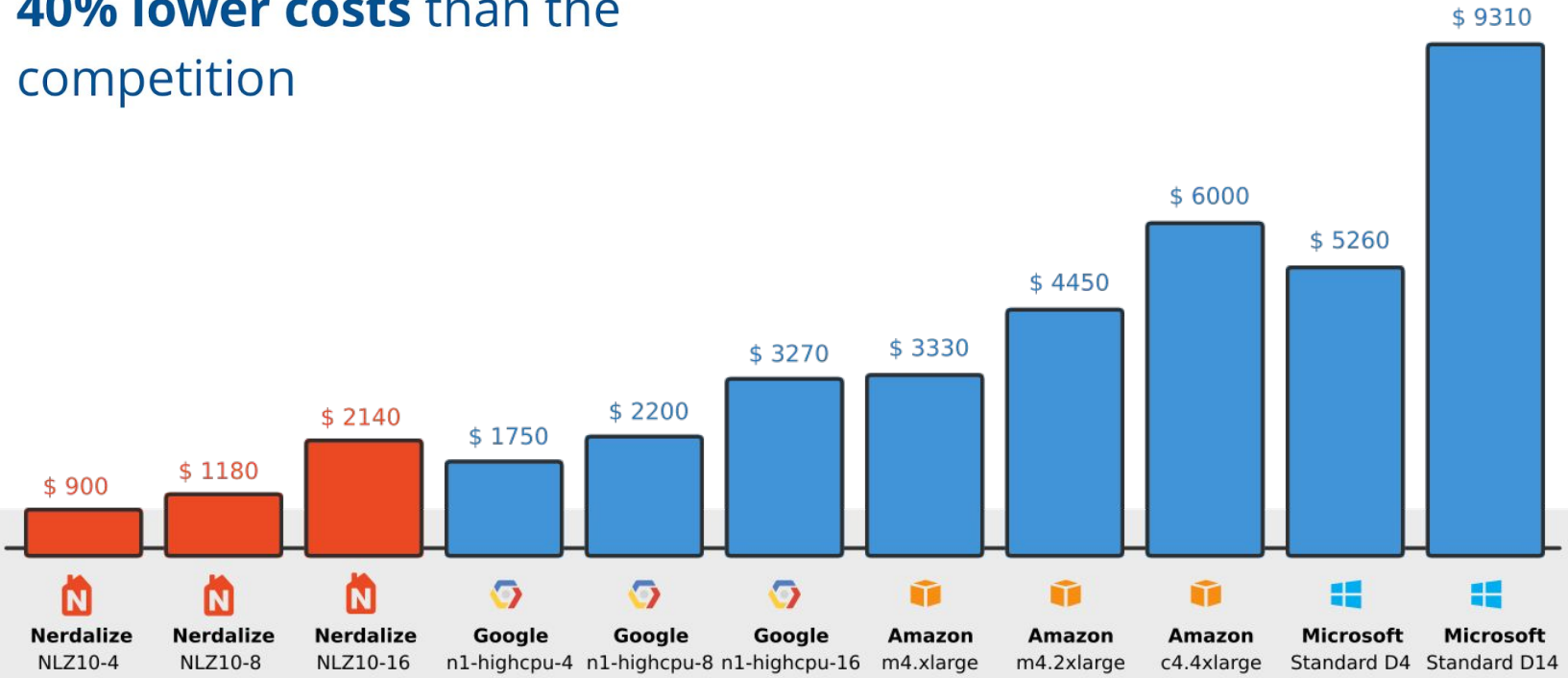
40%
cloud savings

for the cloud user

2 tons / year / home
CO2 savings

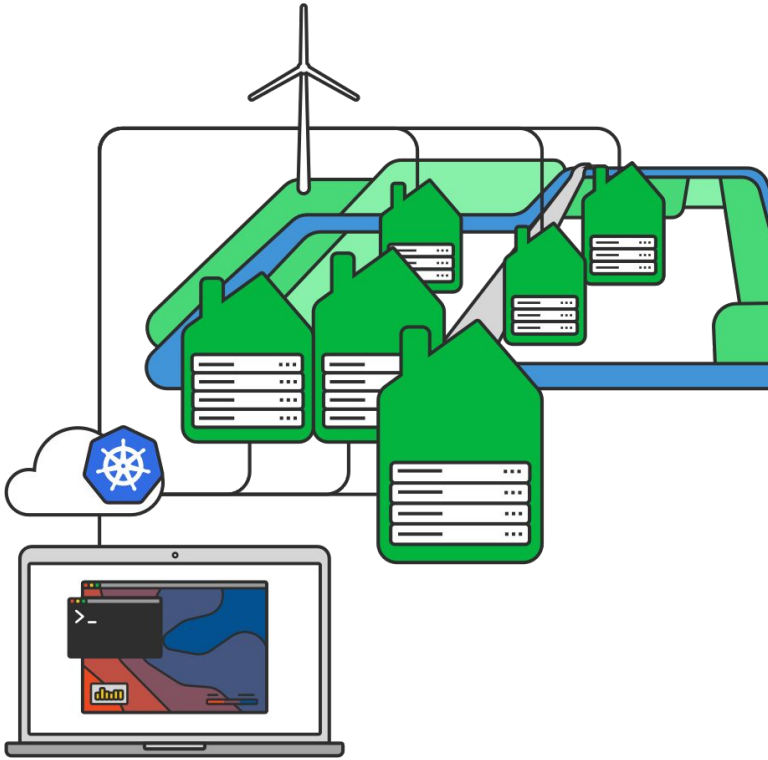
for the world

Nerdalize sells Kubernetes at **40% lower costs** than the competition



Scheduling across homes

We use **Kubernetes** to schedule across homes based on CPU/RAM, but...




Jos
€ 200


Nerdalize
Chantal
55 °C water

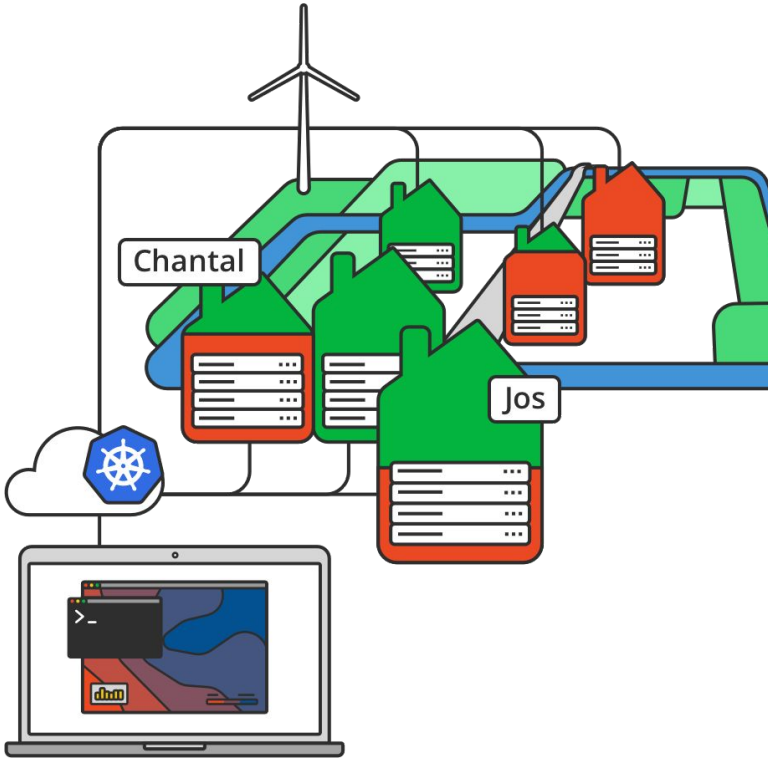
160vCPU
4 servers

10Gb/s
0.15 ms latency

Our challenge:

Scheduling for Heat

Chantal needs heat, **Jos** doesn't. How do we instruct *Kubernetes* to first schedule pods at Chantal and not at Jos?



A Kubernetes Scheduler

*"The **Kubernetes scheduler** is a policy-rich, topology-aware, workload-specific function that significantly impacts availability, performance, and capacity. The scheduler needs to take into account **individual and collective resource requirements**, quality of service requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, deadlines, and so on. Workload-specific requirements will be **exposed through the API** as necessary."*

Kubernetes Scheduling: **In steps**

A scheduler assigns a pod to a node

1. Query the API server for unscheduled *pods*
2. Find a *node* to place each *pod* on
3. For each unscheduled *pod*:
    a. Create a *binding* between the *pod* and the *node*

Basic Scheduling: **In ~10 lines of bash**

```
FIXED_NODE="kubecon-demo"
for PODNAME in $(kubectl --server $SERVER get pods -o json
    | jq '.items[]
    | select(.spec.schedulerName == "nerdalize-scheduler")
    | select(.spec.nodeName == null) | .metadata.name' | tr -d '"'); do
    curl --header "Content-Type:application/json" --request POST --data
        '{"apiVersion":"v1",
          "kind": "Binding",
          "metadata": {"name": "'$PODNAME'"},
          "target": {"apiVersion": "v1", "kind" : "Node", "name":
"'$FIXED_NODE'"}}'
    http://$APISERVER_URL/api/v1/namespaces/default/pods/$PODNAME/binding/
    echo "Assigned $PODNAME to $FIXED_NODE"
done
```

Basic Scheduling: **In ~10 lines of bash**

```
FIXED_NODE="kubecon-demo"
for PODNAME in $(kubectl --server $SERVER get pods -o json
    | jq '.items[]
    | select(.spec.schedulerName == "nerdalize-scheduler")
    | select(.spec.nodeName == null) | .metadata.name' | tr -d '"'); do
    curl --header "Content-Type:application/json" --request POST --data
        '{"apiVersion":"v1",
          "kind": "Binding",
          "metadata": {"name": "'$PODNAME'"},
          "target": {"apiVersion": "v1", "kind" : "Node", "name":
"'$FIXED_NODE'"}}'
    http://$APISERVER_URL/api/v1/namespaces/default/pods/$PODNAME/binding/
    echo "Assigned $PODNAME to $FIXED_NODE"
done
```

Basic Scheduling: **In ~10 lines of bash**

```
FIXED_NODE="kubecon-demo"
for PODNAME in $(kubectl --server $SERVER get pods -o json
    | jq '.items[]
    | select(.spec.schedulerName == "nerdalize-scheduler")
    | select(.spec.nodeName == null) | .metadata.name' | tr -d '"'); do
    curl --header "Content-Type:application/json" --request POST --data
        '{"apiVersion":"v1",
          "kind": "Binding",
          "metadata": {"name": "'$PODNAME'"},
          "target": {"apiVersion": "v1", "kind" : "Node", "name":
"'$FIXED_NODE'"}}'
    http://$APISERVER_URL/api/v1/namespaces/default/pods/$PODNAME/binding/
    echo "Assigned $PODNAME to $FIXED_NODE"
done
```

Basic Scheduling: **In ~10 lines of bash**

```
FIXED_NODE="kubecon-demo"
for PODNAME in $(kubectl --server $SERVER get pods -o json
    | jq '.items[]
    | select(.spec.schedulerName == "nerdalize-scheduler")
    | select(.spec.nodeName == null) | .metadata.name' | tr -d '"'); do
    curl --header "Content-Type:application/json" --request POST --data
        '{"apiVersion":"v1",
          "kind": "Binding",
          "metadata": {"name": "'$PODNAME'"},
          "target": {"apiVersion": "v1", "kind" : "Node", "name":
"'$FIXED_NODE'"}}'
    http://$APISERVER_URL/api/v1/namespaces/default/pods/$PODNAME/binding/
    echo "Assigned $PODNAME to $FIXED_NODE"
done
```

Basic Scheduling: **In ~10 lines of bash**

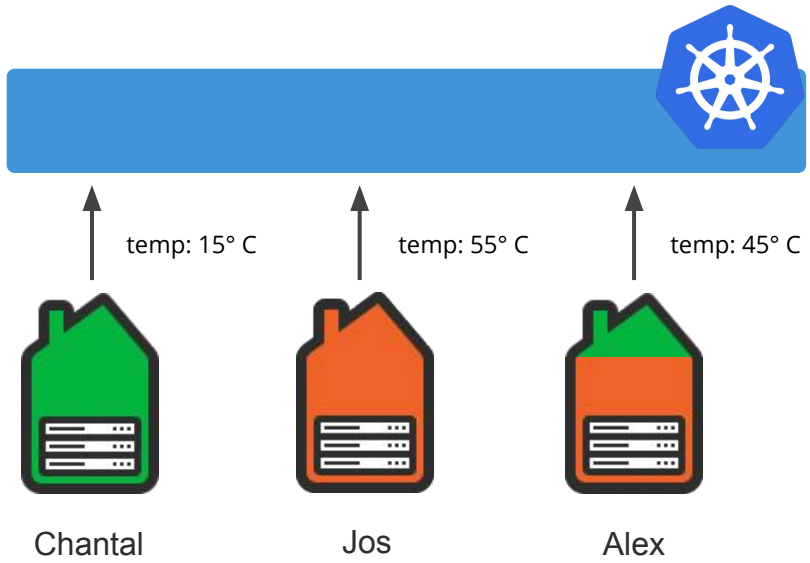
```
FIXED_NODE="kubecon-demo"
for PODNAME in $(kubectl get pods -o json
    | jq '.items[]
    | select(.spec.schedulerName == "nerdalize-scheduler")
    | select(.spec.nodeName == null) | .metadata.name' | tr -d '"'); do
    curl --header "Content-Type:application/json" --request POST --data
        '{"apiVersion":"v1",
          "kind": "Binding",
          "metadata": {"name": "'$PODNAME'"},
          "target": {"apiVersion": "v1", "kind" : "Node", "name":
"'$FIXED_NODE'"}}'
    http://$APISERVER_URL/api/v1/namespaces/default/pods/$PODNAME/binding/
    echo "Assigned $PODNAME to $FIXED_NODE"
done
```

Boris Mattijssen

Software Engineer at Nerdalize

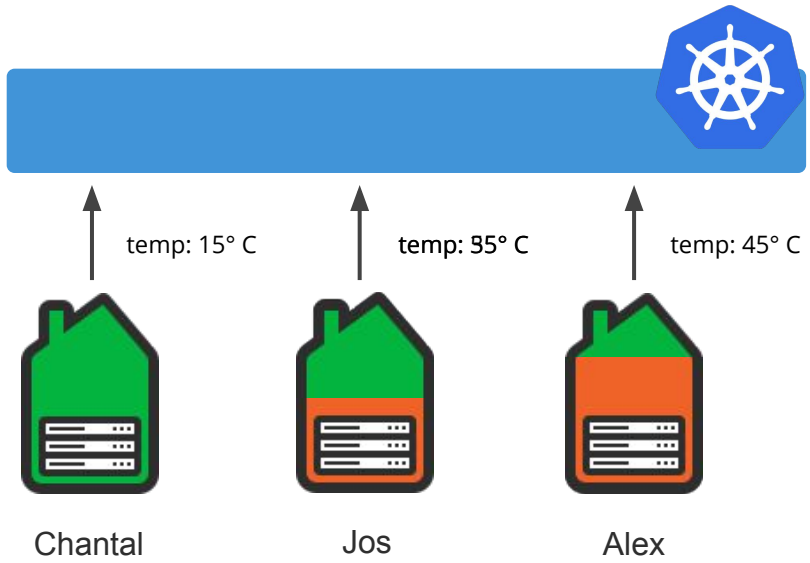
Advanced Scheduling: **Considering heat**

1. Every home has a ***boiler (reservoir)*** filled with water
2. The ***temperature*** of this water is measured…
3. and reported to Kubernetes as a ***node annotation***



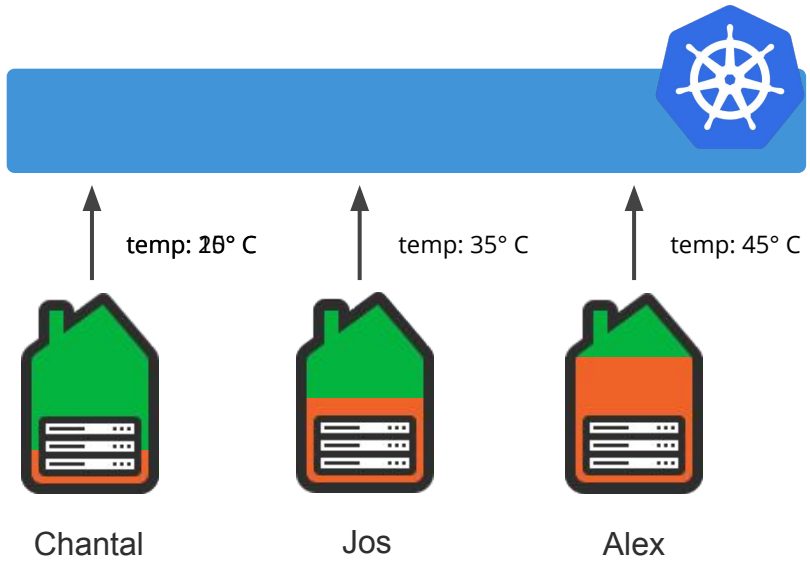
Advanced Scheduling: **Considering heat**

- When homes use warm water, this value ***decreases***



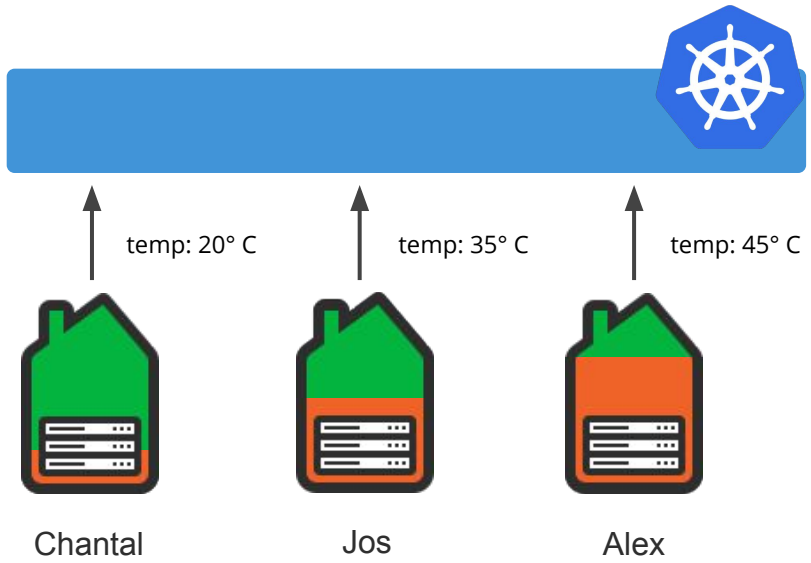
Advanced Scheduling: **Considering heat**

- When homes use warm water, this value ***decreases***
- When pods run on a node, the boiler temperature will ***increase***

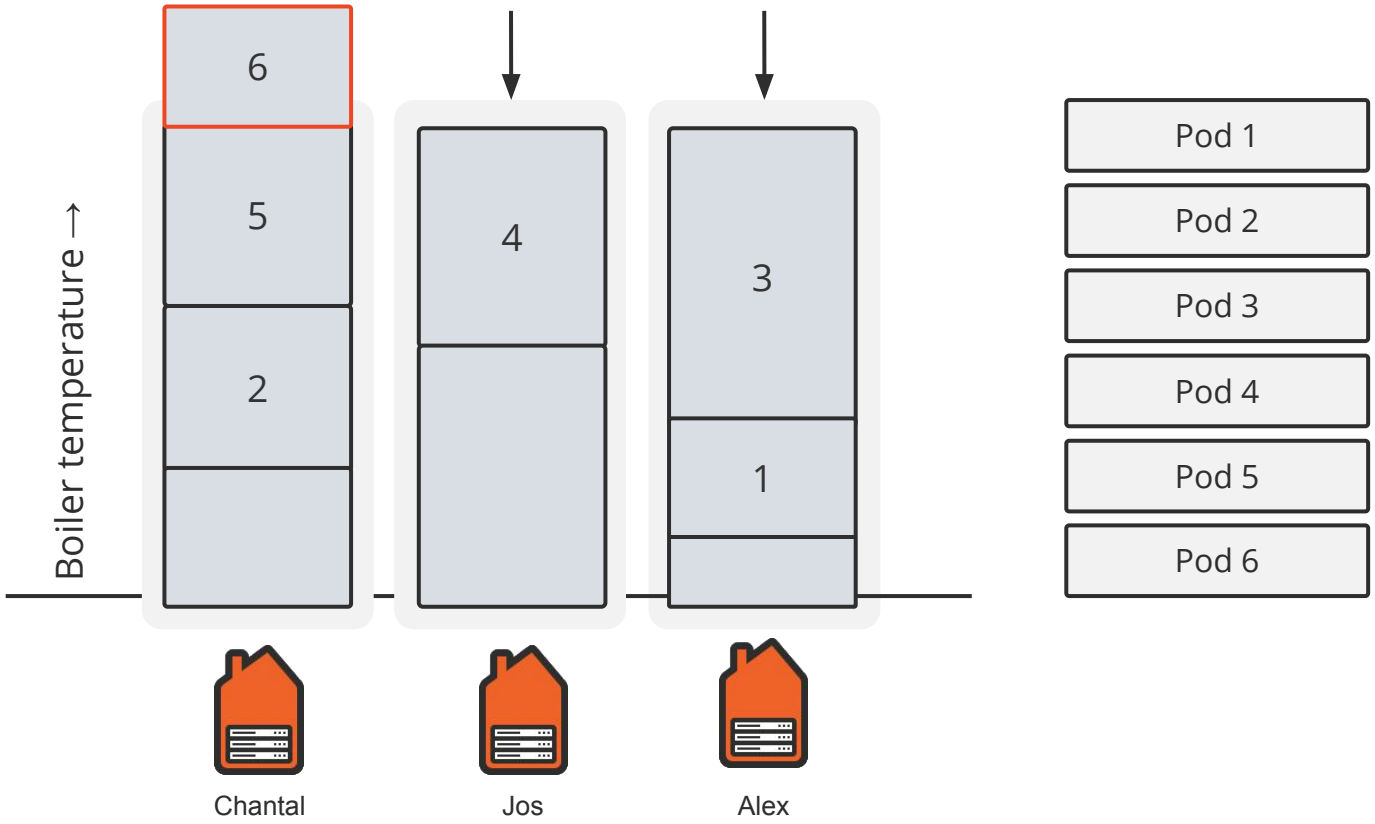


Advanced Scheduling: **Considering heat**

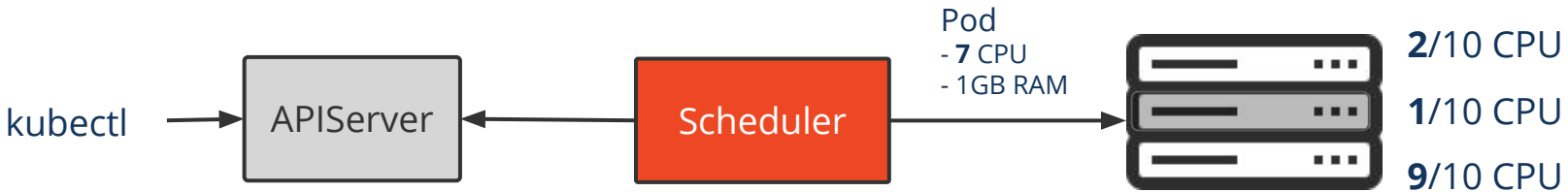
- When homes use warm water, this value ***decreases***
- When pods run on a node, the boiler temperature will ***increase***
- The ***node annotation*** is updated with the new temperature



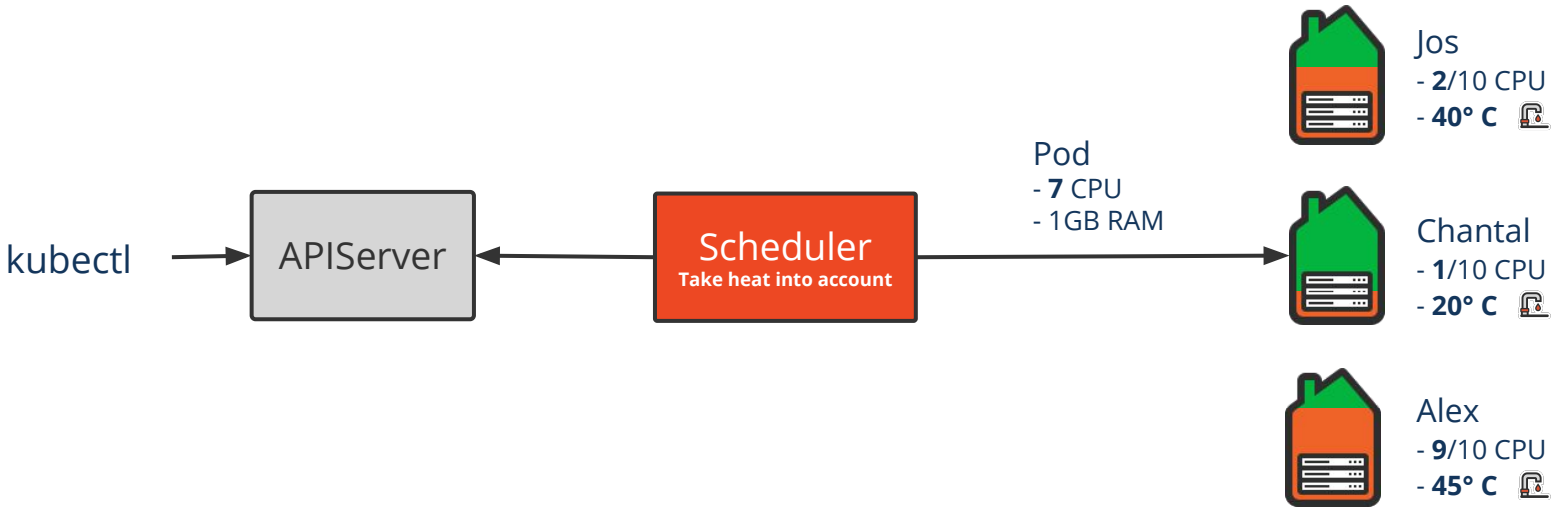
Advanced Scheduling: **Considering heat**



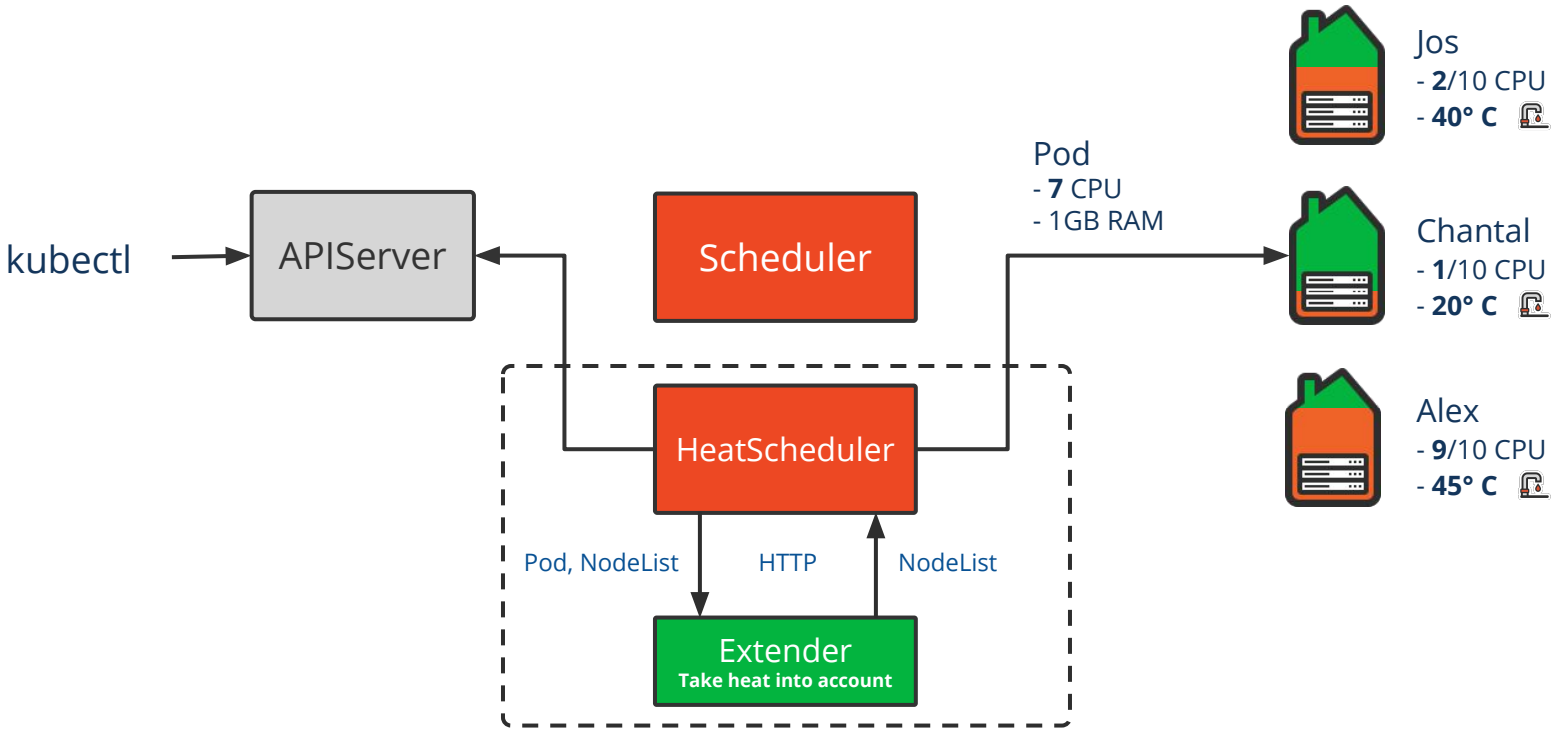
Advanced Scheduling: **Extending Kubernetes**



Advanced Scheduling: **Extending Kubernetes**



Advanced Scheduling: **Extending Kubernetes**



Extending Kubernetes: **HeatScheduler Configuration**

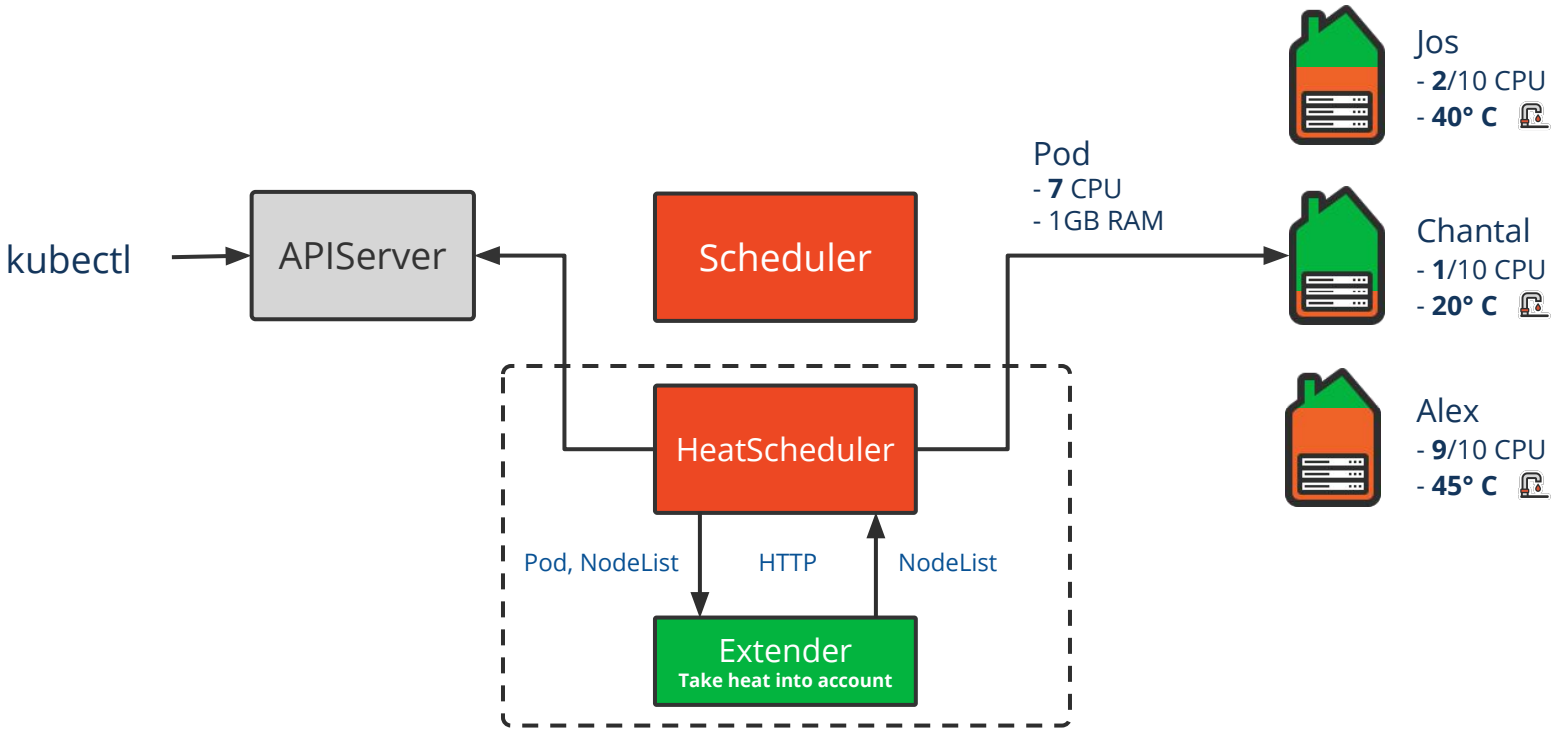
heat-scheduler.yaml

```
apiVersion: apps/v1
kind: Deployment
..
spec:
 ..
   spec:
     containers:
     - command:
       - /usr/local/bin/kube-scheduler
       - --address=0.0.0.0
       - --scheduler-name=heat-scheduler
       - --policy-config-file=policy-config-file.json
       image: nerdalize/heat-scheduler
..
```

policy-config-file.json

```
{
 "kind" : "Policy",
 "apiVersion" : "v1",
 "predicates" : [{
   "name": "PodFitsResources"
 }],
 "extenders": [{
   "urlPrefix":
"http://heat-scheduler-extdr.scheduler",
   "apiVersion": "v1",
   "filterVerb": "filter",
   "enableHttps": false
 }]
}
```

Advanced Scheduling: **Extending Kubernetes**



Extending Kubernetes: **Extender in Code**

```
func handler(resp http.ResponseWriter, req *http.Request) {
 // decode request body.
 dec := json.NewDecoder(req.Body)
 received := &api.ExtenderArgs{}
 dec.Decode(received)
```

1

2

3

Extending Kubernetes: **Architecture Overview**

namespace: scheduler

pod: heat-scheduler

- Core kube-scheduler binary
- Config file that points to http://heat-scheduler-extdr.scheduler

service: heat-scheduler-extdr

pod: heat-scheduler-extender

- HTTP endpoint
- Scheduling logic

Experiment: **Scenario**

We **heat 50 homes**
on average **4 people per home**

Experiment: **Scenario**

We **heat 50 homes**
on average **4 people per home**

Compare default **Scheduler vs HeatScheduler**

Experiment: **Scenario**

We **heat 50 homes**
on average **4 people per home**

Compare default **Scheduler vs HeatScheduler**

Children shower
15 minutes

Experiment: **Scenario**

We **heat 50 homes**
on average **4 people per home**

Compare default **Scheduler vs HeatScheduler**

Children shower
15 minutes

Boiler warms up
1.5 hours

Experiment: **Scenario**

We **heat 50 homes**
on average **4 people per home**

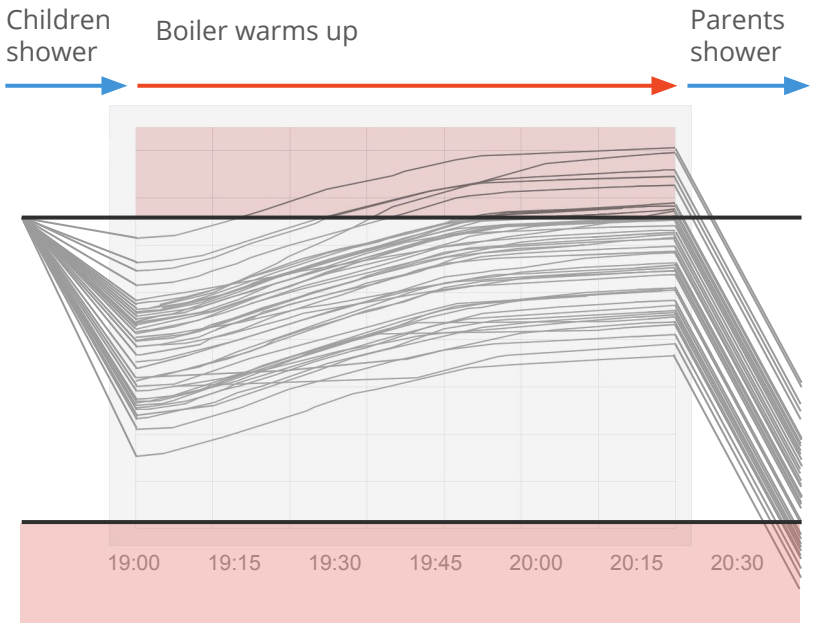
Compare default **Scheduler vs HeatScheduler**

Children shower
15 minutes

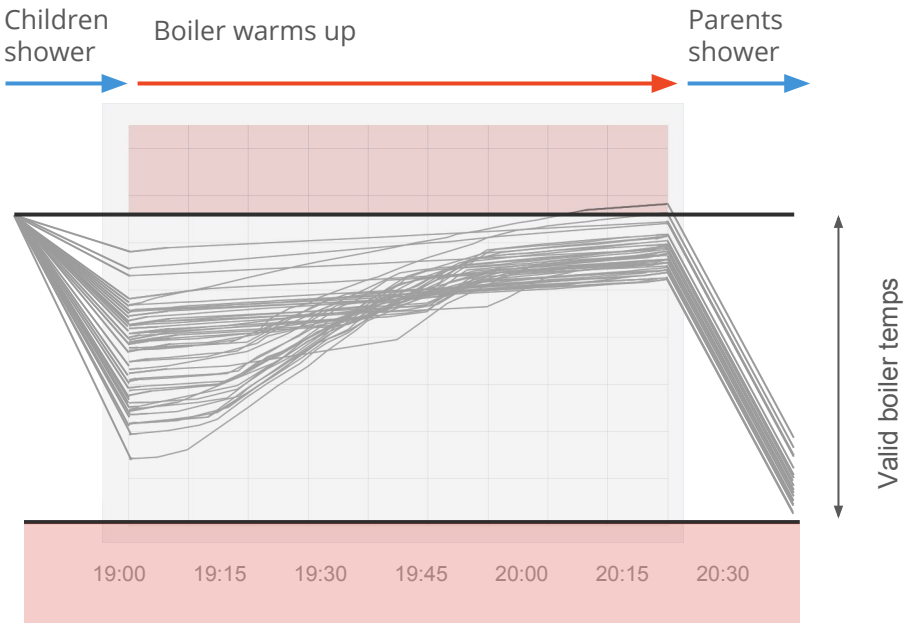
Boiler warms up
1.5 hours

Parents shower
15 minutes

Experiment: **Results**



Default Scheduler



Heat Scheduler

10%

Additional CO_2 emissions reduced

Robert Carosi

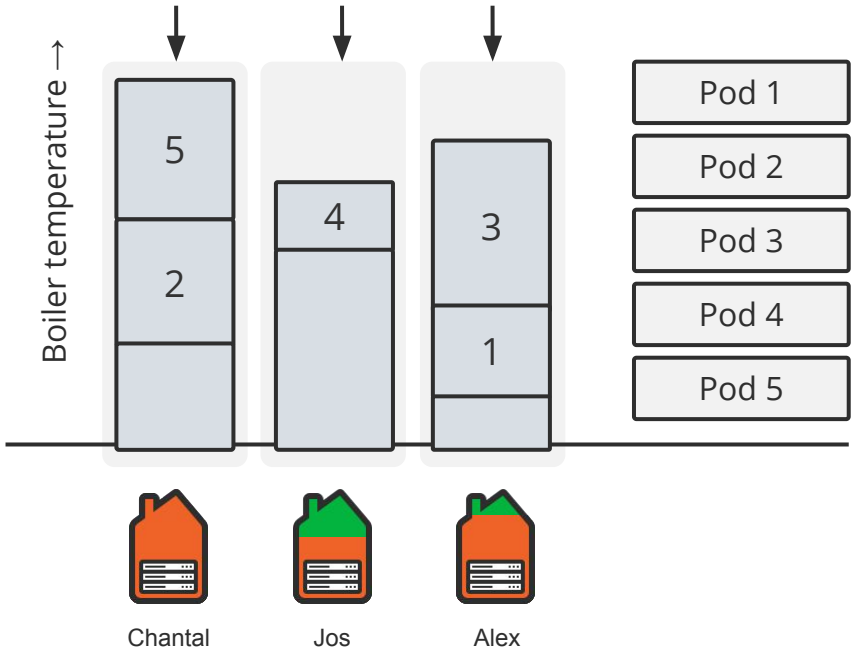
Graduate at Spotify

Advanced Scheduling: **Demo**

Advanced Scheduling: **Future Possibilities**

Current scheduling strategy:

1. Minimum boiler temperature



Advanced Scheduling: **Future Possibilities**

Current scheduling strategy:

1. Minimum boiler temperature

Possible strategies:

2. Forecast boiler heat up

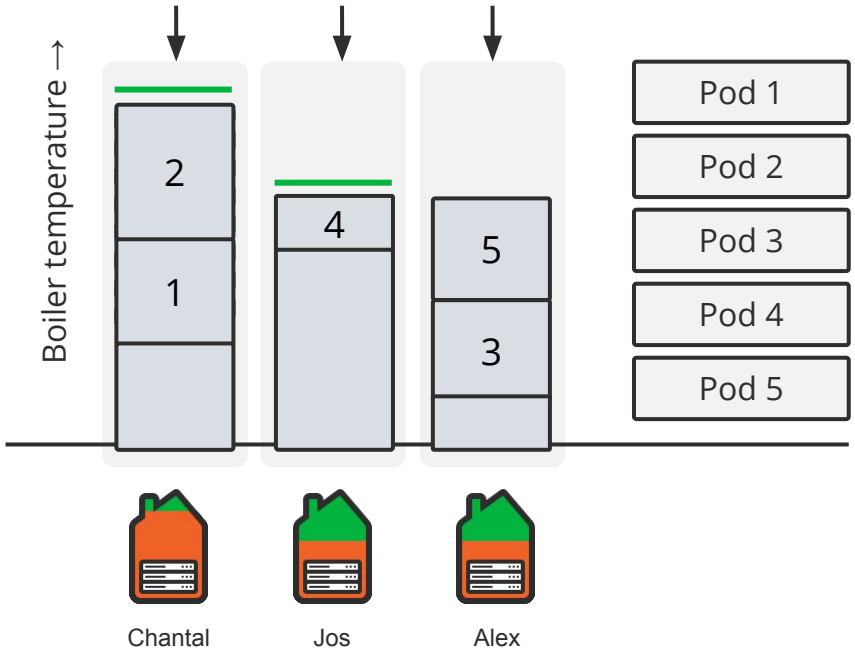
Advanced Scheduling: **Future Possibilities**

Current scheduling strategy:

1. Minimum boiler temperature

Possible strategies:

2. Forecast boiler heat up
3. Forecast heat need



Future possibilities: **You can use this too**

1

```
func handler(resp http.ResponseWriter, req *http.Request) {
 // decode request body.
 dec := json.NewDecoder(req.Body)
 received := &api.ExtenderArgs{}
 dec.Decode(received)
```

2

```
 // find minimum
 min := math.MaxFloat64
 for _, node := range received.Nodes.Items {
  min = math.Min(min, node.Annotations["nerdalize/temp"])
 }

 // find node belonging to minimum heat value
 var output []api.Node
 for _, node := range received.Nodes.Items {
  if min == node.Annotations["nerdalize/temp"] {
   output = []api.Node{node}
  }
 }
```

3

```
 // write response
 enc := json.NewEncoder(resp)
 enc.Encode(&api.ExtenderFilterResult{
  Nodes: api.NodeList{
   Items: output,
  },
 })
}
```

Future possibilities: **You can use this too**

```
// 1. schedule by hardware type
output := findSSDNodes(received.Nodes.Items)

// 2. schedule on free instances first
output := findFreeInstances(received.Nodes.Items)

// 3. schedule where the data is
output := findNodeWithData(received.Nodes.Items, pod.datasetId)

// 4. the world is yours
output := yourBusinessLogicHere()
```

2

Wrap up: **What we discussed**

Nerdalize builds a datacenterless cloud which:

- Saves homeowners €200 / year
- Reduces 2 Tons CO_2 emissions / year / home
- Allows offering Kubernetes at 40% lower costs

With the Nerdalize use case we talked about:

- A basic scheduler in bash
- How to implement a heat-aware extender
- The ease of adapting this to you own business-logic
- An experiment that showed an additional 10% CO_2 emission reduction

Take a look at our code

And feel free to commit: github.com/nerdalize

Join the Nerdalize mission

Help grow our team of Nerds
Check out and share our job openings: careers.nerdalize.com

Heat the Netherlands with your Kubernetes pods
Free credits, every core counts: https://www.nerdalize.com/kubecon

Ask the Nerdalize KubeCon 2018 crew
Or send your questions to cloud@nerdalize.com