

# Building a Cloud Toolkit

Jay DesLauriers



# The Research Centre for Parallel Computing @ UoW

Projects in Distributed Computing, from Grids to Cloud (to Fog/Edge)

Well-funded by EU/UK Research Grants (>£2.5mil since 2015)

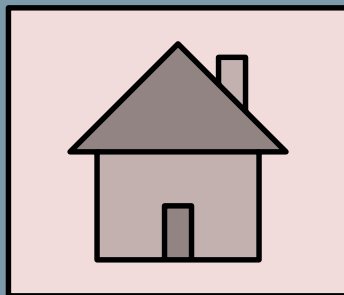
- EDGeS: Enabling Desktop Grids for e-Science (2008)
- EDGI: European Desktop Grid Initiative (2010)
- VENUS-C: Virtual multidisciplinary environments using Cloud Infrastructures (2012)
- CloudSME: Cloud based Simulation Platform for Manufacturing & Engineering (2013)
- COLA: **Cloud Orchestration at the Level of Application (2017)**
- ASCLEPIOS: Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare (2018)

# Why Cloud?

It's disruptive. Compute now available "as-a-Service"

No upfront cost for hardware or software licenses

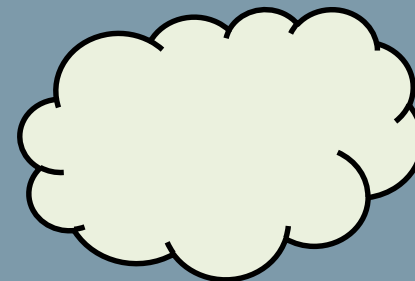
No operating or maintenance cost for local IT infrastructure



**On-Premise**

Capital expense model

££££



**Cloud**

Pay-as-you-go model

££

# Reality-Check

Take-up still relatively low for research applications & by small business

Vendor lock-in: going multi-cloud is expensive, complex or both

Application-level auto-scaling is limited

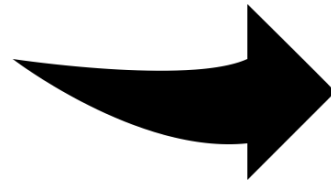
Issues of security, privacy and trust

# Project COLA

EU Horizon2020 Programme for Research & Innovation

33 months, 14 partners, 6 countries

Secure, cloud agnostic application-level auto-scaling  
to encourage cloud uptake



# MiCADO

Microservice-based Cloud Application-level Dynamic Orchestrator



Terraform



Kubernetes



Prometheus



Interface



Security

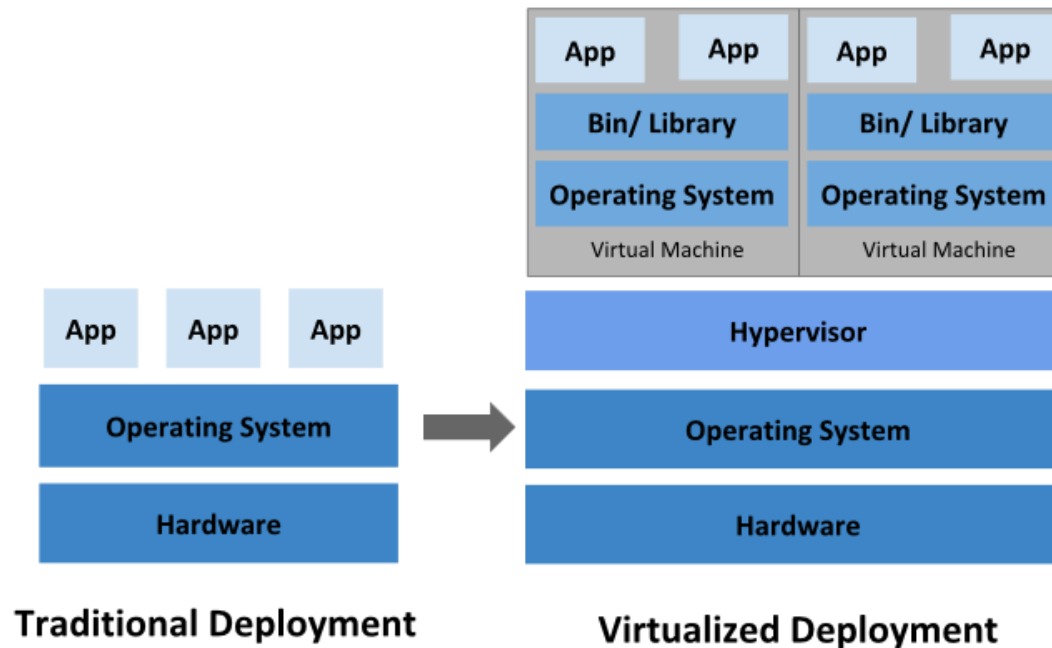


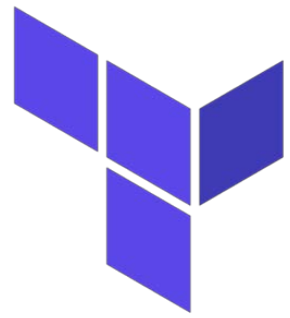
Scaling

# Cloud Orchestration

## Infrastructure-as-a-Service (IaaS)

- Provisioning virtual machines from a cloud service provider (CSP)





# Manual Cloud Provisioning

The screenshot shows the top navigation bar of the AWS console. It includes the AWS logo, a 'Services' dropdown menu, and a 'Resource Groups' dropdown menu. Below the navigation bar, a horizontal breadcrumb trail shows seven steps: '1. Choose AMI', '2. Choose Instance Type', '3. Configure Instance', '4. Add Storage', '5. Add Tags', '6. Configure Security Group', and '7. Review Instance Launch'. Step 2 is currently selected and highlighted with an orange underline.

## Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, applications, and data) that you want to use to launch an Amazon EC2 instance.

## Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run a wide range of applications, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types.

The screenshot shows the top navigation bar of the AWS console. It includes the AWS logo, a 'Services' dropdown menu, and a 'Resource Groups' dropdown menu. Below the navigation bar, a horizontal breadcrumb trail shows three steps: '1. Choose AMI', '2. Choose Instance Type', and '3. Configure Instance'. Step 3 is currently selected and highlighted with an orange underline.

## Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances.

The screenshot shows the top navigation bar of the AWS console. It includes the AWS logo, a 'Services' dropdown menu, and a 'Resource Groups' dropdown menu. Below the navigation bar, a horizontal breadcrumb trail shows six steps: '1. Choose AMI', '2. Choose Instance Type', '3. Configure Instance', '4. Add Storage', '5. Add Tags', and '6. Configure Security Group'. Step 6 is currently selected and highlighted with an orange underline.

## Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, to allow unrestricted access to the Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select an existing security group.

The screenshot shows the top navigation bar of the AWS console. It includes the AWS logo, a 'Services' dropdown menu, and a 'Resource Groups' dropdown menu. Below the navigation bar, a horizontal breadcrumb trail shows seven steps: '1. Choose AMI', '2. Choose Instance Type', '3. Configure Instance', '4. Add Storage', '5. Add Tags', '6. Configure Security Group', and '7. Review Instance Launch'. Step 7 is currently selected and highlighted with an orange underline.

## Step 7: Review Instance Launch

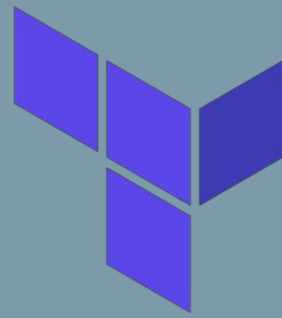
Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

**⚠ Improve your instances' security.** Your instances may be accessible from any IP address. You can also open additional ports in your security groups.

### Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.





# Terraform

## Infrastructure-as-Code

```
provider "aws" {  
  region = "us-west-2"  
}  
  
data "aws_ami" "ubuntu" {  
  most_recent = true  
  
  filter {  
    name = "name"  
    values = ["ubuntu/images/hvm-ssd/ubuntu-t  
  }  
}  
  
resource "aws_instance" "web" {  
  ami = "${data.aws_ami.ubuntu.id}"  
  instance_type = "t2.micro"  
  
  tags = {
```

```
TERMINAL > terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...
```

```
The refreshed state will be used to calculate this plan, but will  
persisted to local or remote state storage.
```

```
-----  
An execution plan has been generated and is shown below.
```

```
Resource actions are indicated with the following symbols:
```

```
+ create
```

```
Terraform will perform the following actions:
```

```
# aws_instance.web will be created
```

```
+ resource "aws_instance" "web" {
```

```
  + ami
```

```
    = "ami-0bac6fc47ad07c5f5"
```



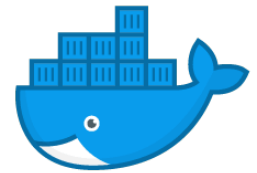
# Terraform

## Infrastructure Management

- Provisions
- Maintains
- Destroys

- Scales
- Self-healing

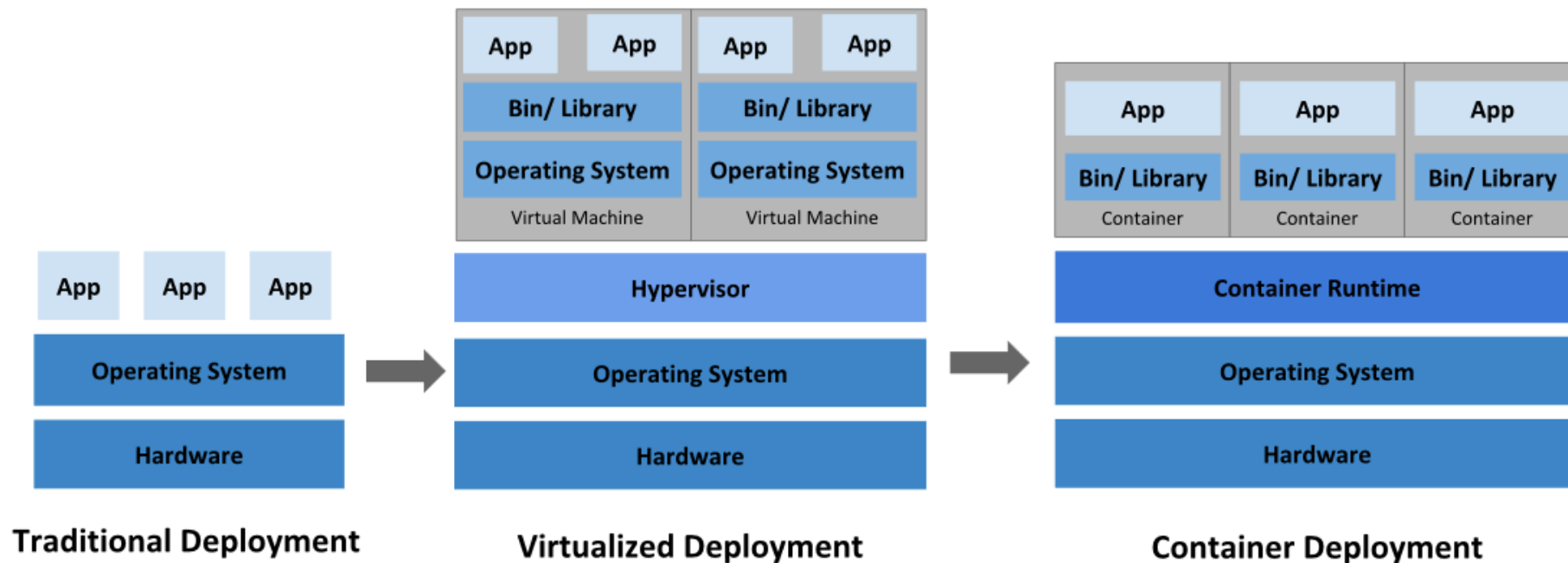


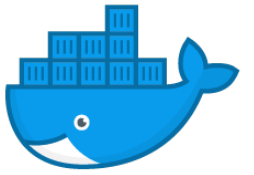


# Container Orchestration

## Application containers

- Lightweight OS-virtualisation
- Application packaging for portable, reusable software





# Container Orchestration

```
TERMINAL > docker run busybox cal -j
```

```
          December 2019
Su  Mo  Tu  We  Th  Fr  Sa
335 336 337 338 339 340 341
342 343 344 345 346 347 348
349 350 351 352 353 354 355
356 357 358 359 360 361 362
363 364 365
```

```
TERMINAL > docker run -d busybox sleep 60
211685b29840d758974795a662b14c1d6df807ec792faed90fc84b0557b84e5b
```

```
TERMINAL > docker ps
```

CONTAINER ID	IMAGE	COMMAND
68c958637f70	busybox	"sleep 60"



# Kubernetes

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes']
```

```
TERMINAL > vim kube-test.yaml
kubectl apply -f kube-test.yaml
pod/myapp-pod created
TERMINAL > kubectl get po
NAME          READY   STATUS    RESTARTS   AGE
myapp-pod    1/1     Running   0           6s
TERMINAL > █
```





# Kubernetes

## Container Management Across a Cluster of Nodes (VMs)

- Deploys
- Maintains
- Destroys
- Auto-scaling
- Self-healing
- Rolling updates and rollbacks CI/CD



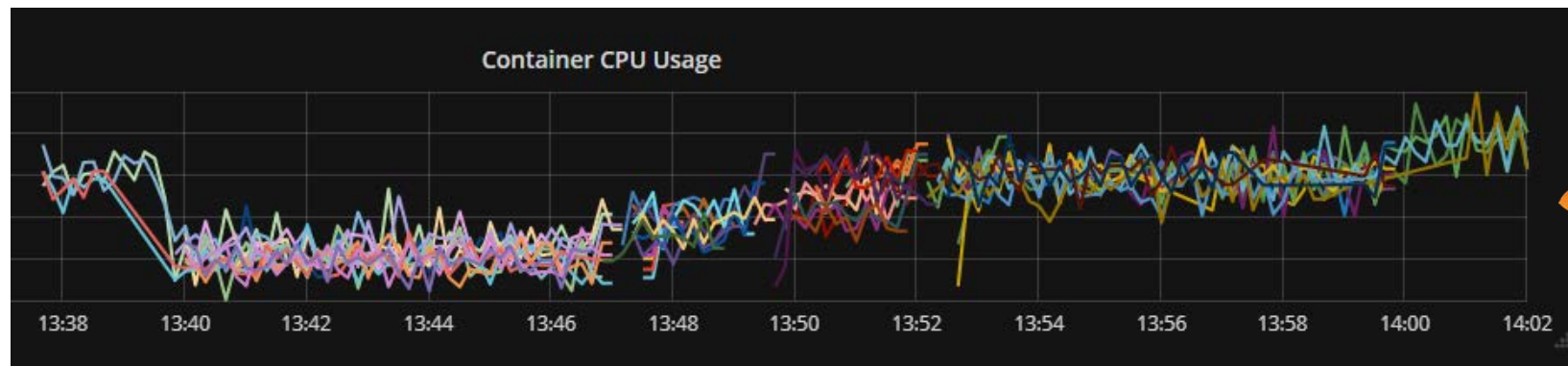
# Prometheus

A monitoring & alerting system

Pull-style metric collection

- Resource usage of containers / virtual machines (CPU, Memory, etc...)
- Custom data exported by applications (latency, requests served)

Alerting based on those metrics



# Terraform, Kubernetes & Prometheus for Research



## Some good things:

- Open-Source
- Community
- Extensions
  - Kubernetes (Kubeflow)
  - Prometheus Exporters (DBs)
  - Terraform Modules (Sagewatch, BigQuery)

## Could-be-better things:

- High overall complexity
  - Deploying, configuring, integrating
- Vendor lock-in encouraged
- Limited scope for auto-scaling





# MiCADO

Microservice-based Cloud Application-level Dynamic Orchestrator



Cloud  
Orchestration



Container  
Orchestration



Monitoring

UNIVERSITY OF  
WESTMINSTER

Interface



BALASYS

Security



MTA  
SZTAKI

Scaling

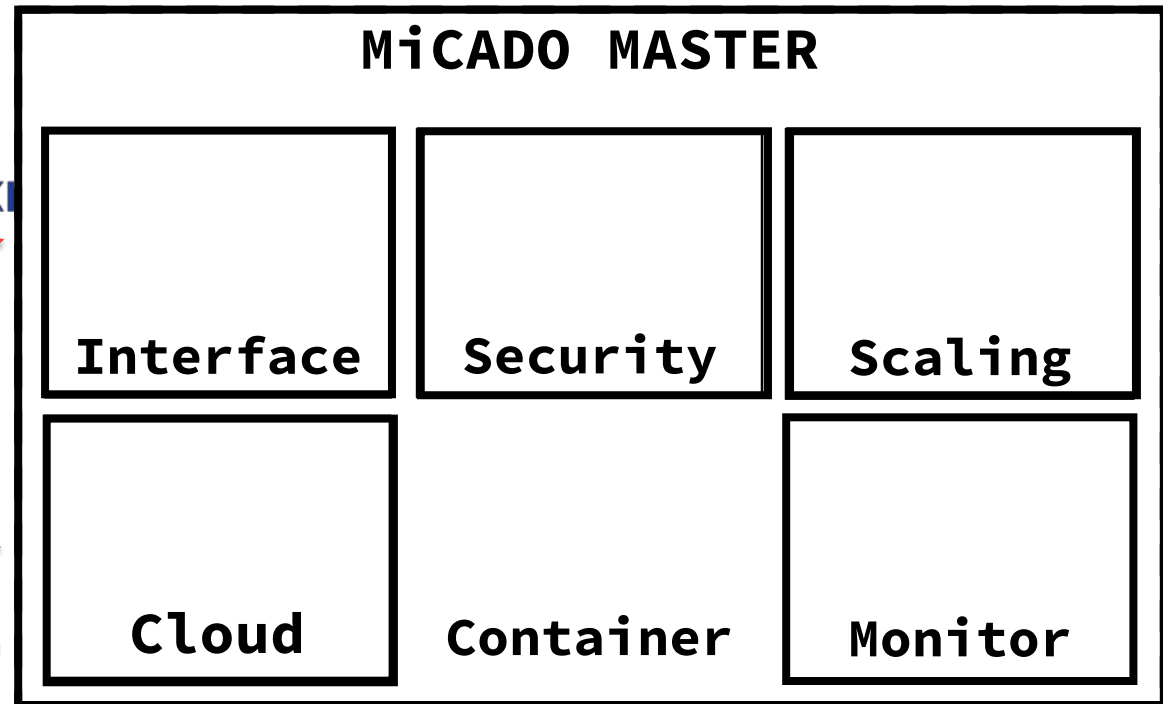


# Ansible

Declarative configuration management automation framework

Deploys and configures MiCADO microservices

ansible  
playbook



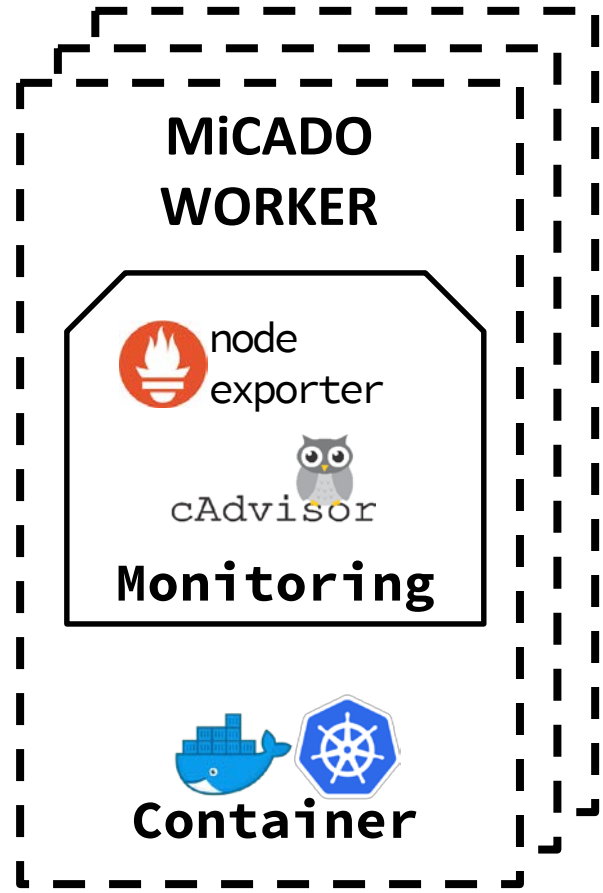
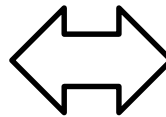
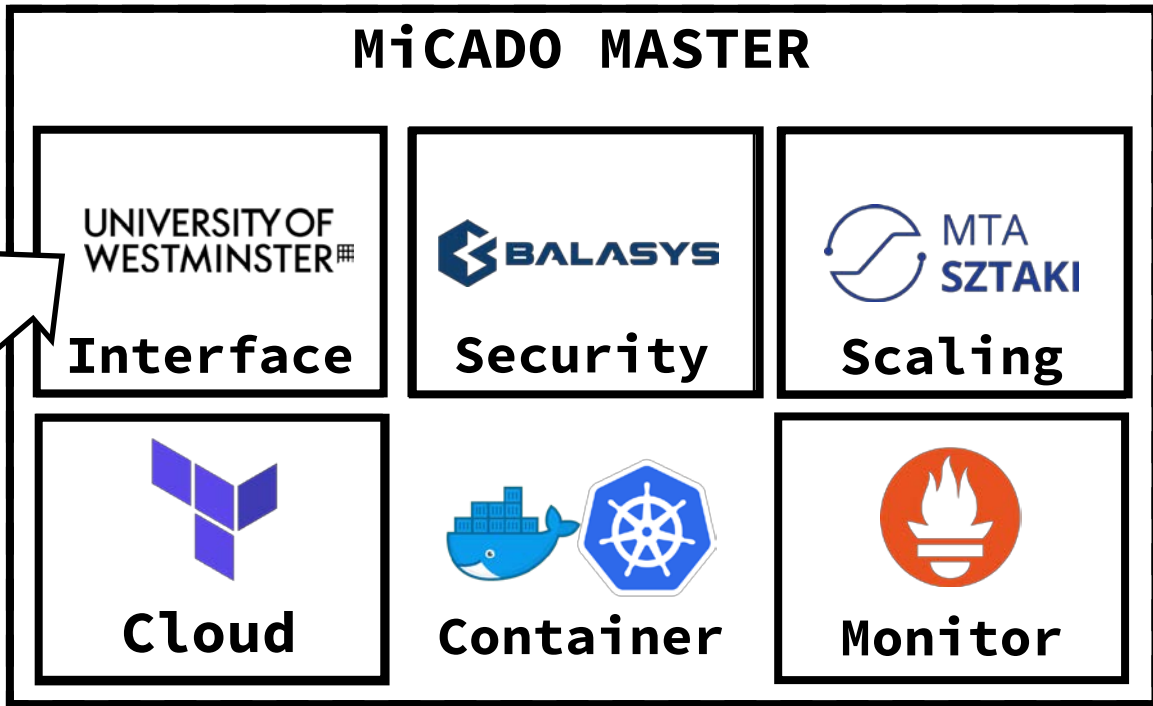
# MiCADO

**TOSCA**

*virtual machines*  
4cores, 4GB

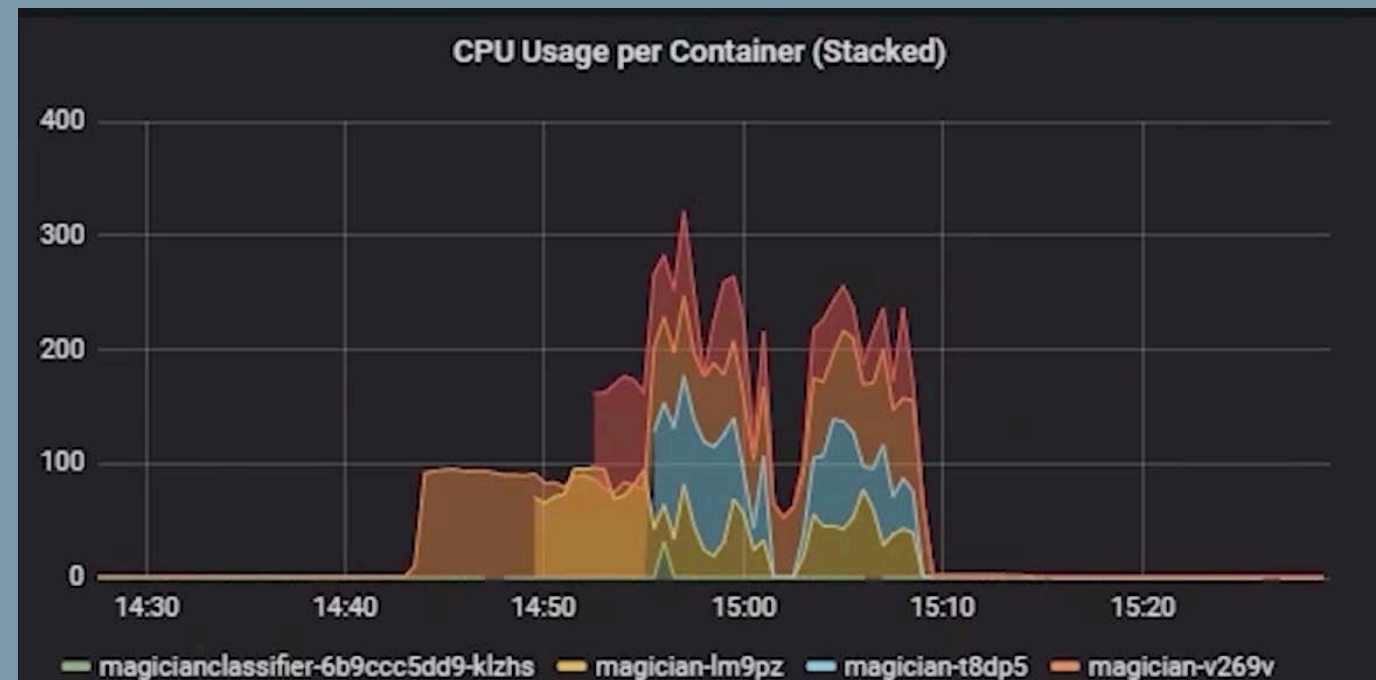
*containers*  
docker.hub/image

*policy*  
scale at > 75% CPU



# Social Media Analytics Use-Case

- Resource intensive services
  - Typically CPU/memory -bound apps/services
  - Containers & underlying VMs scale to meet demand

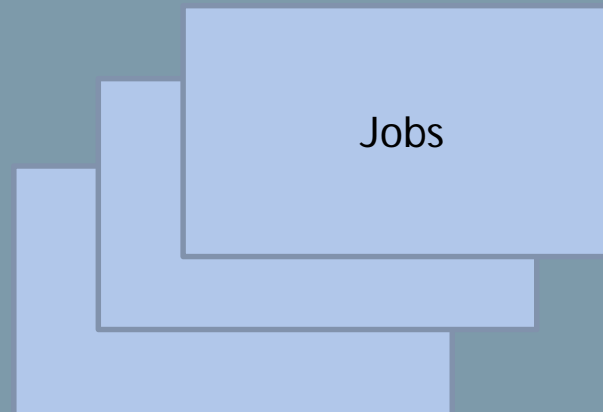


# Simulation & Modelling Use-Case

- Multi-job, deadline constrained experiments
  - Typically batch/parameter sweep jobs
  - Containers/VMs scale to complete jobs by deadline

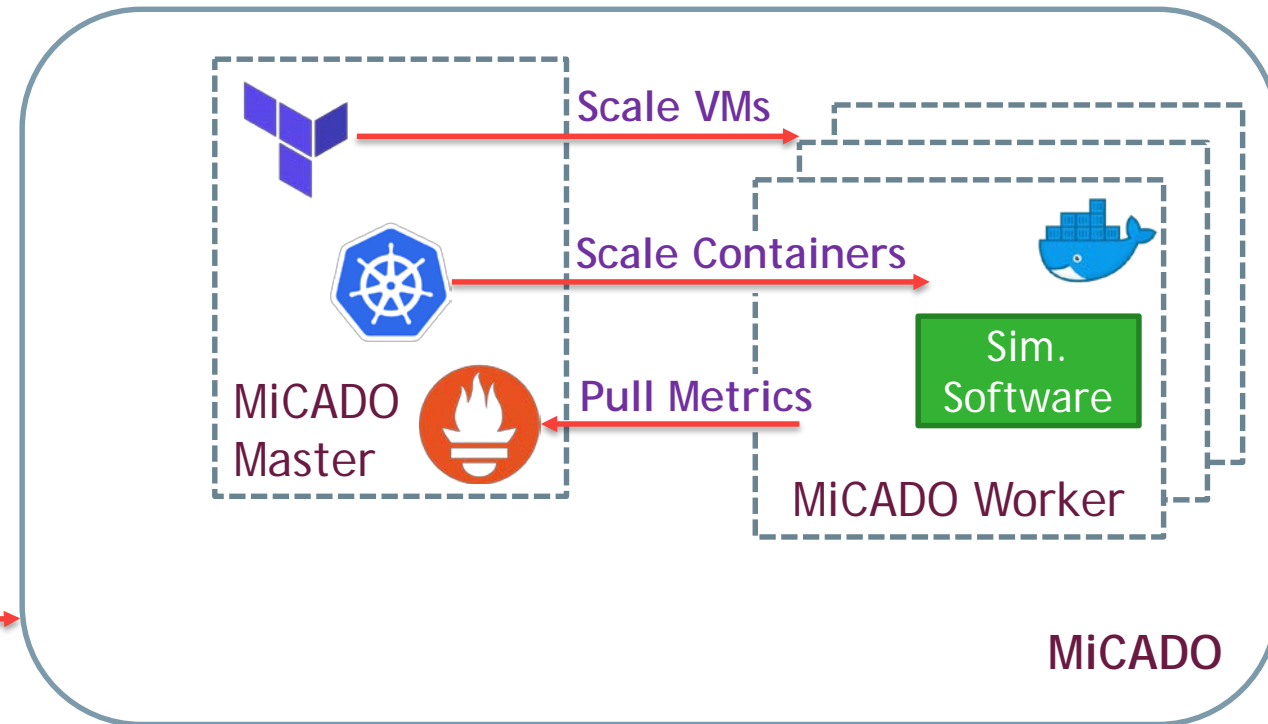


**Brunel**  
University  
London



# Insert Queue Here

Where do we put the jobs?  
How do we execute them



# jQueuer

## Asynchronous Distributed Task Queue

- Celery.py

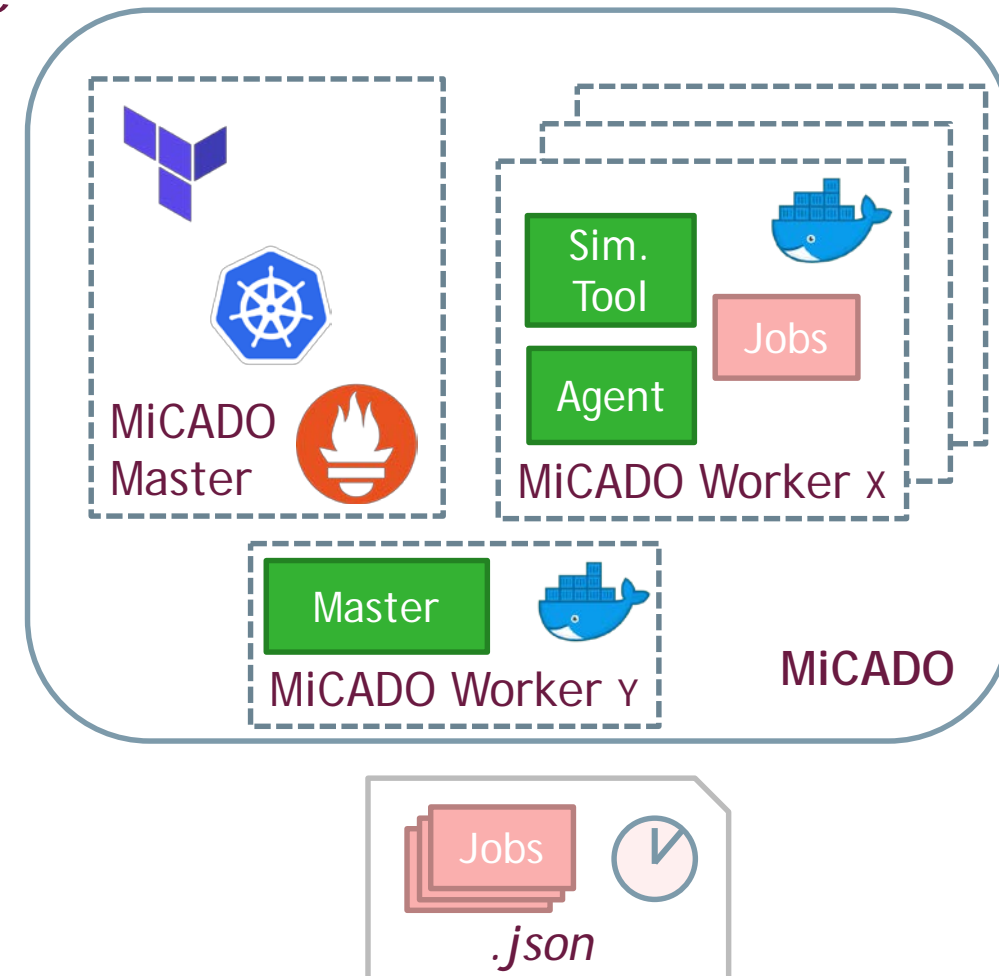
### Master Component

- The queue
- Metric Generation
- Frontend for submission

### Agent Component

- Runs alongside experiment tool
- Fetches jobs from Master
- Executes jobs in container

- JSON input
  - Jobs & Deadline



# Deadline-based auto-scaling

Calculates containers/VMs required to complete jobs by deadline

Uses jQueuer metrics:

- Queue length
- Jobs completed
- Jobs remaining
- Time elapsed
- Average job length
- Time to deadline

Cloud resources are scaled up/down by MiCADO

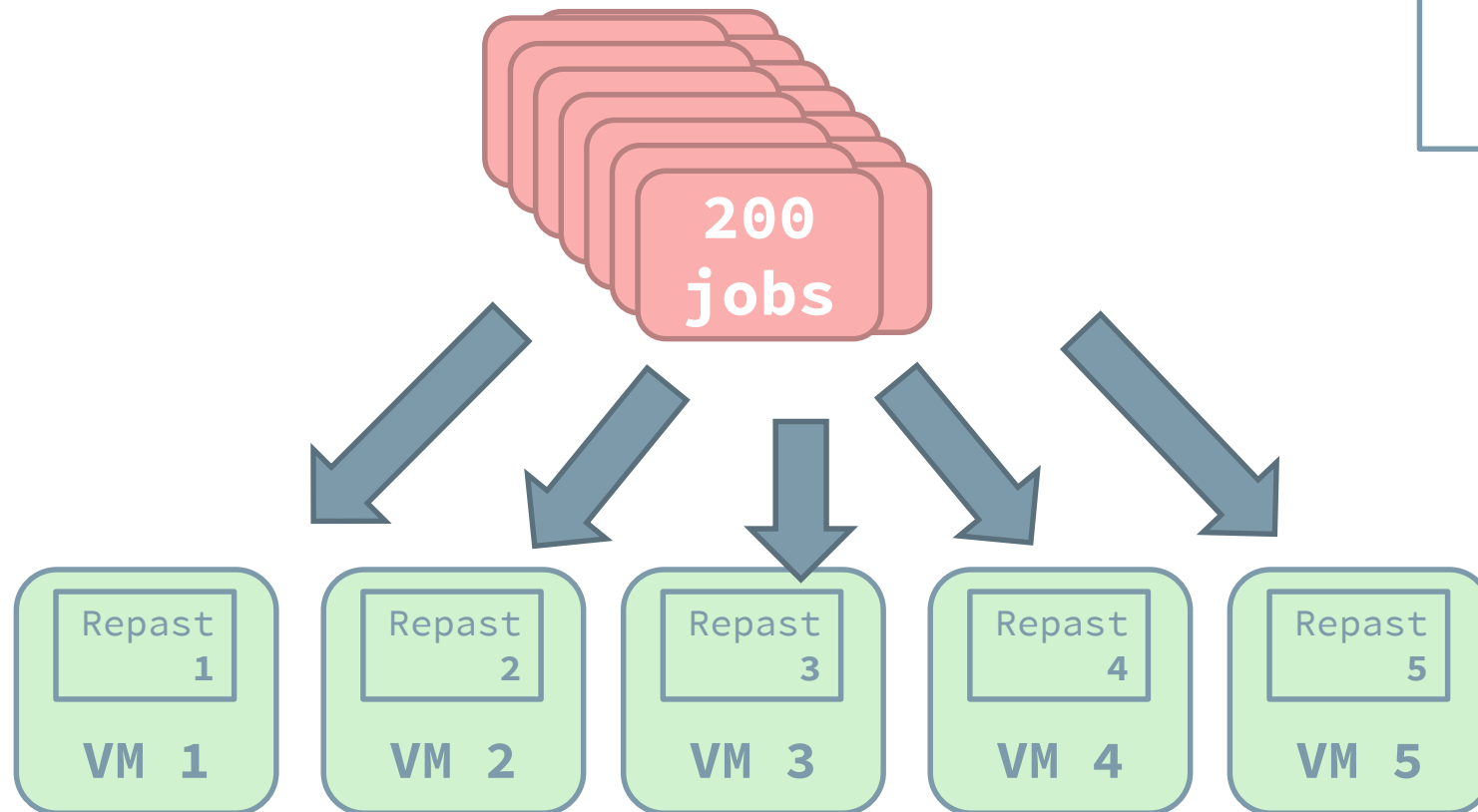


# THE EXPERIMENT

- Agent-based simulation
  - Repast Symphony
- Three agents
  - Infected
  - Susceptible
  - Recovered
- Simulate movement & interaction of agents in an environment to determine effects of one group on another



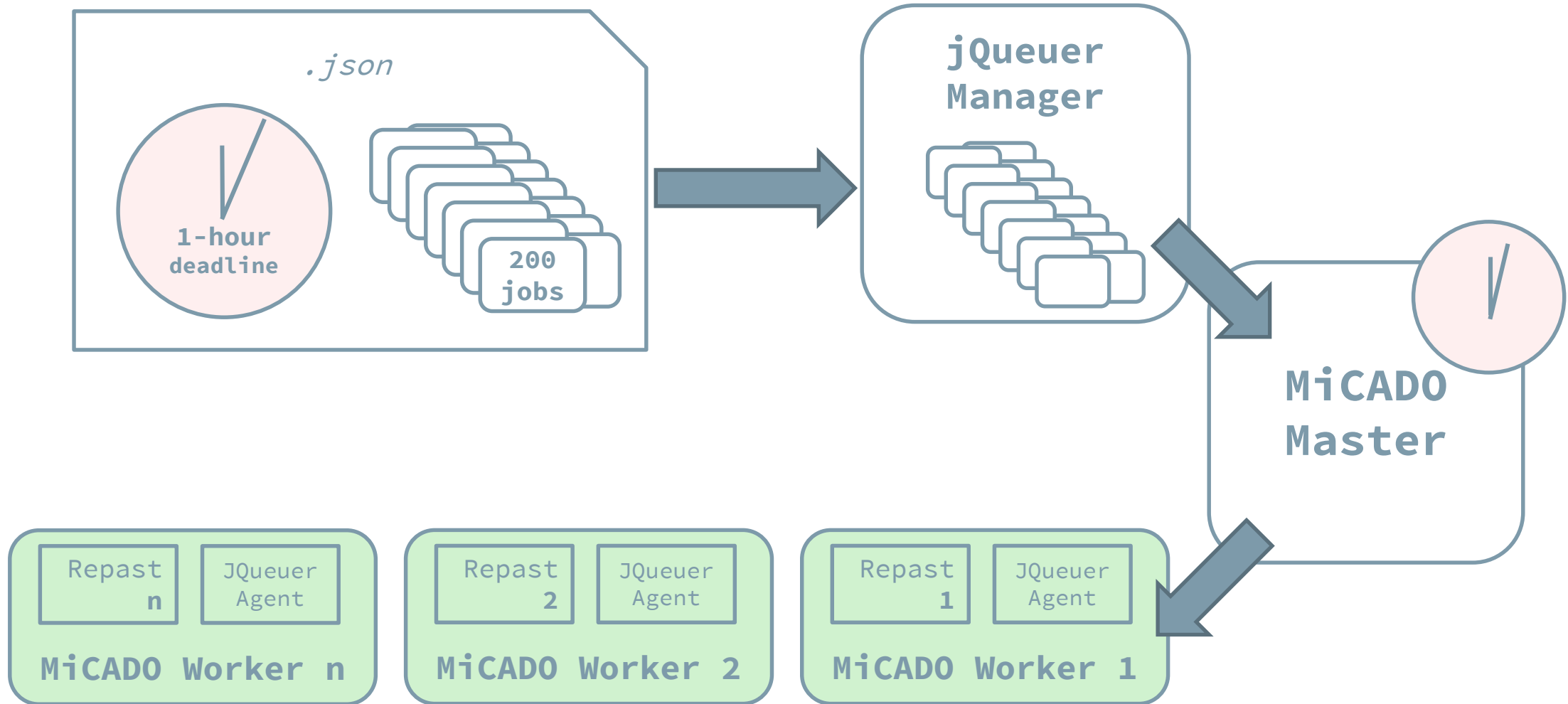
## Calculating a Baseline: Manual allocation

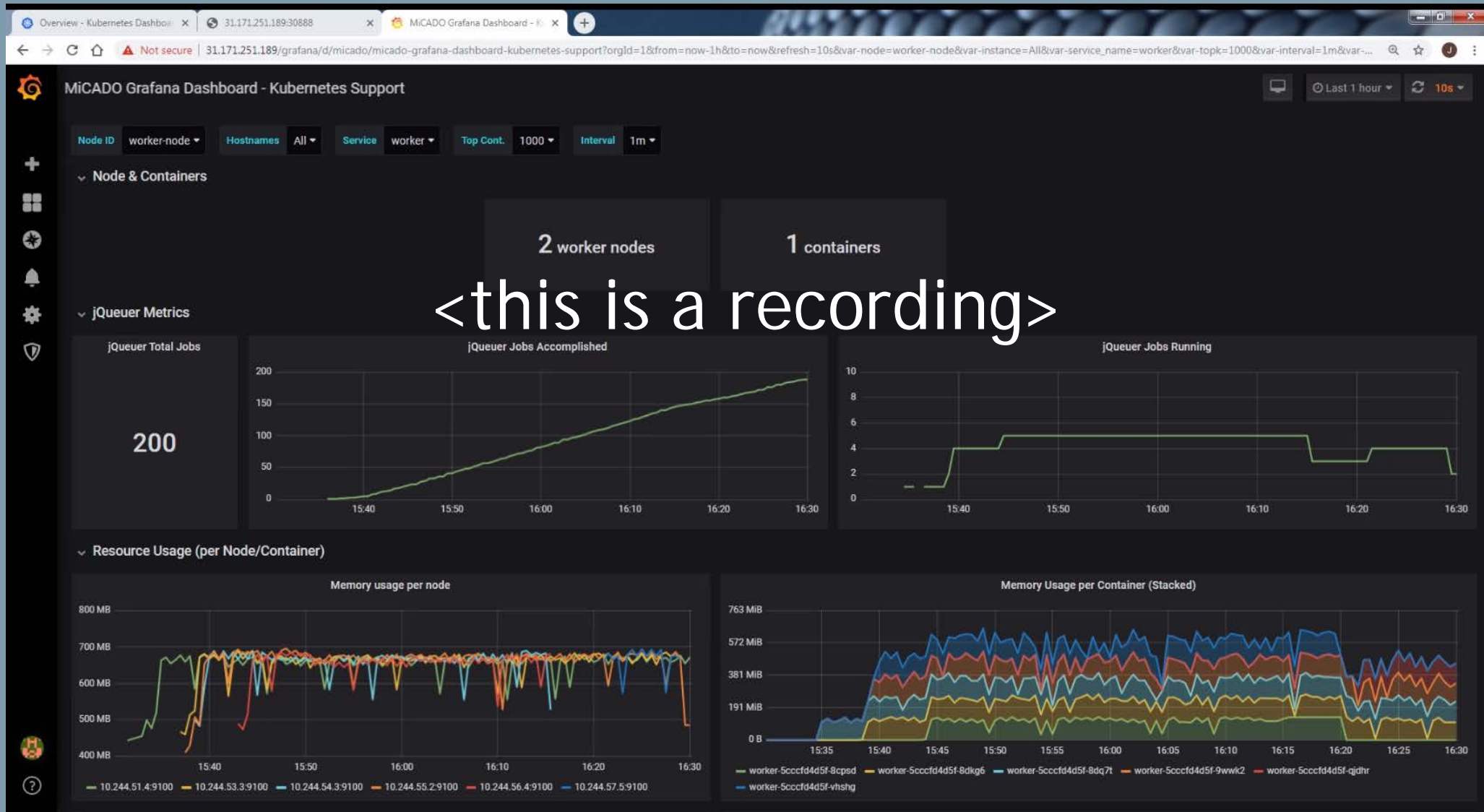


Equal distribution to five virtual machines running Repast in container (40 jobs per VM)



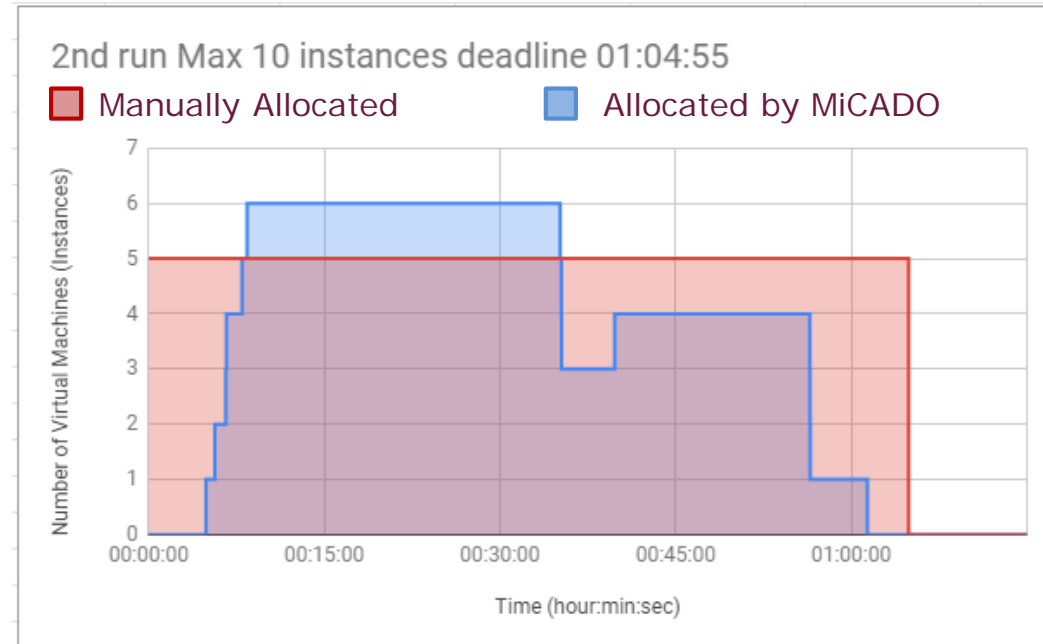
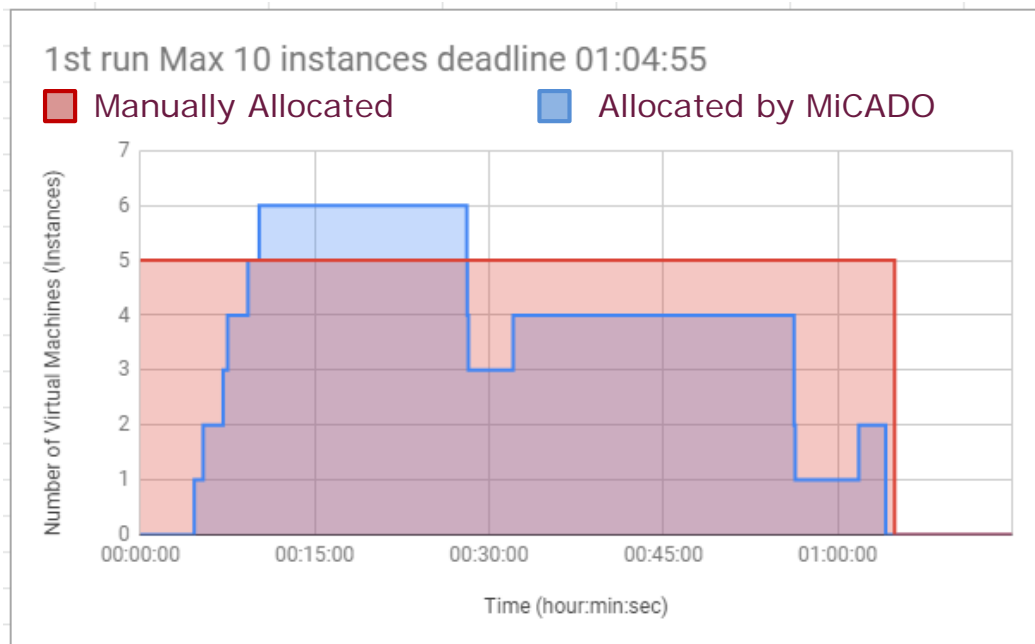
## Using MiCADO: Dynamic allocation & auto-scaling





## Results

Dynamic allocation of variable length jobs  
results in a better use of cloud resources



**5 VMs**

Manual  
allocation  
(baseline)

**3.86 VMs**

Dynamic  
allocation  
(MiCADO)

# Cast (in order of appearance)

**Terraform** [terraform.io](https://terraform.io)

**Kubernetes** [kubernetes.io](https://kubernetes.io)

**Prometheus** [prometheus.io](https://prometheus.io)

**Ansible** [ansible.io](https://ansible.io)

**MiCADO** [micado-scale.eu](https://micado-scale.eu)

**jQueuer** [doi.org/10.1016/j.future.2019.05.062](https://doi.org/10.1016/j.future.2019.05.062)

# Thanks!

Jay DesLauriers  
[j.deslauriers@westminster.ac.uk](mailto:j.deslauriers@westminster.ac.uk)