Just enough Docker

Modernize your traditional applications and more

RAMIRO BERRELLEZA | ARCHITECT | @RBERRELLEZA

PABLO CHICO DE GUZMÁN | SENIOR SOFTWARE ENGINEER | @CHICO_DE_GUZMAN

Opinions are our own, they don’t represent the views of our employers

Agenda

Why containers?

Get your containers online

i2kit

Legacy Application

Manual infrastructure

Physical server.

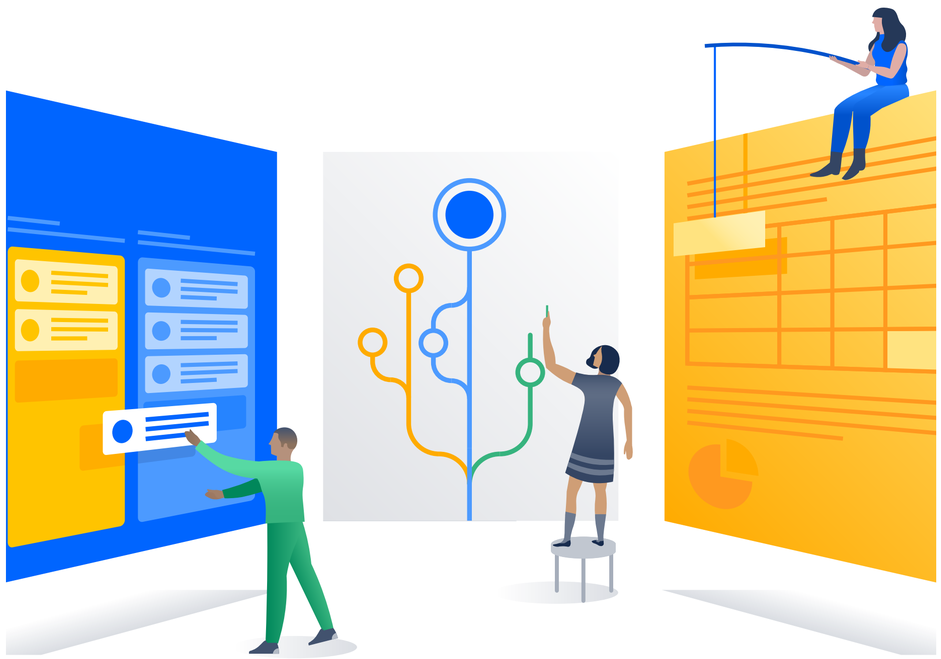
Manual operations

Deployment, upgrade and troubleshooting is all done manually by the application administrator.

Not Replicable

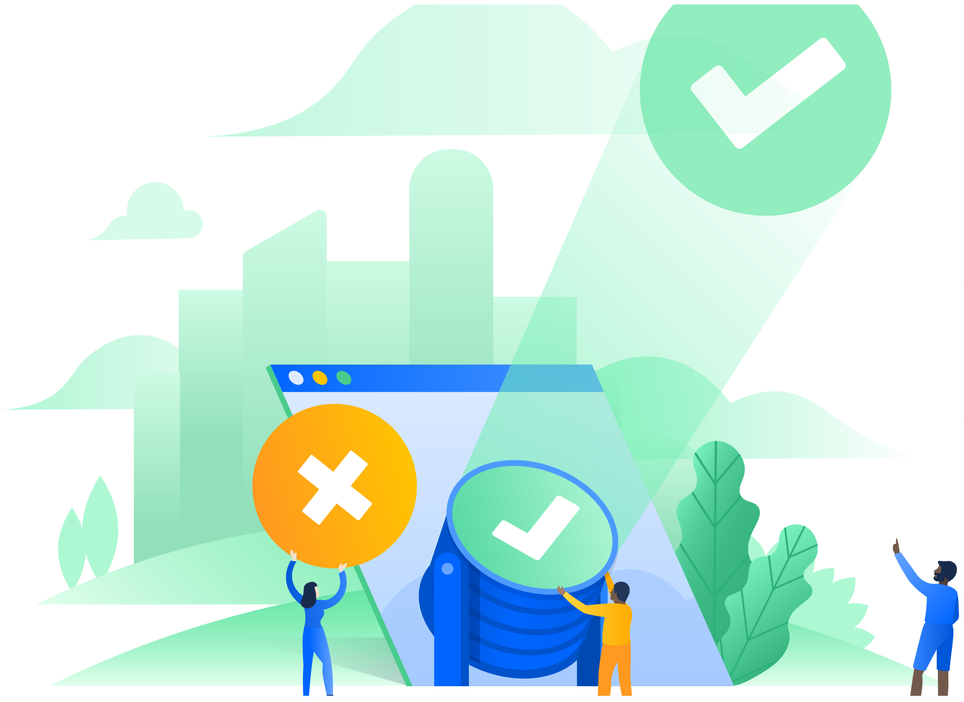
Can’t create new environments automatically.

Cloud Application



Automated infra

Infrastructure and application can be deployed automatically via scripts (e.g. cloud formation templates).



Replicable

Application can be deployed to different environments, automatically tested via Jenkins, Bamboo or similar CI/CD system.

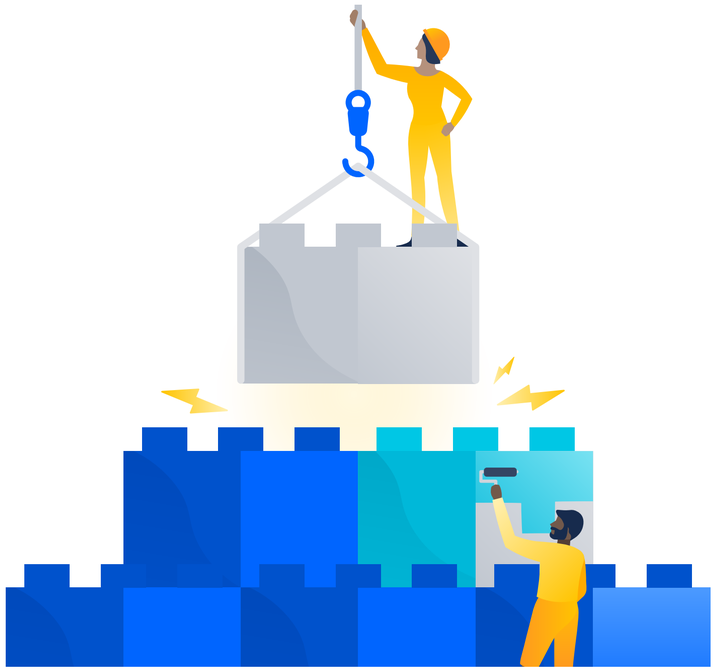
Highly available

Leverages different cloud services (Load balancer, scaling groups, dynamic DNS) to always be online.

Not Portable

Works in my machine!

Containerized Application



Container Runtime

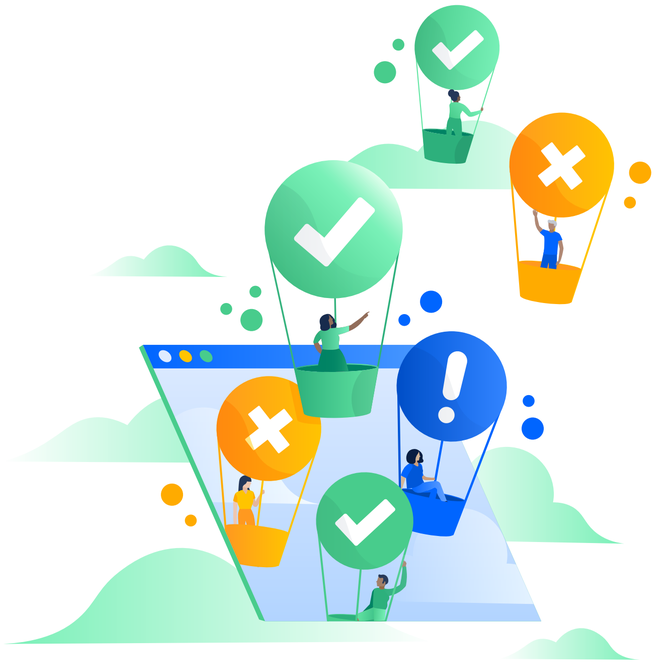
Unifying dev, ci, stage and prod environments.

Container Images

Can be easily distributed across machines.

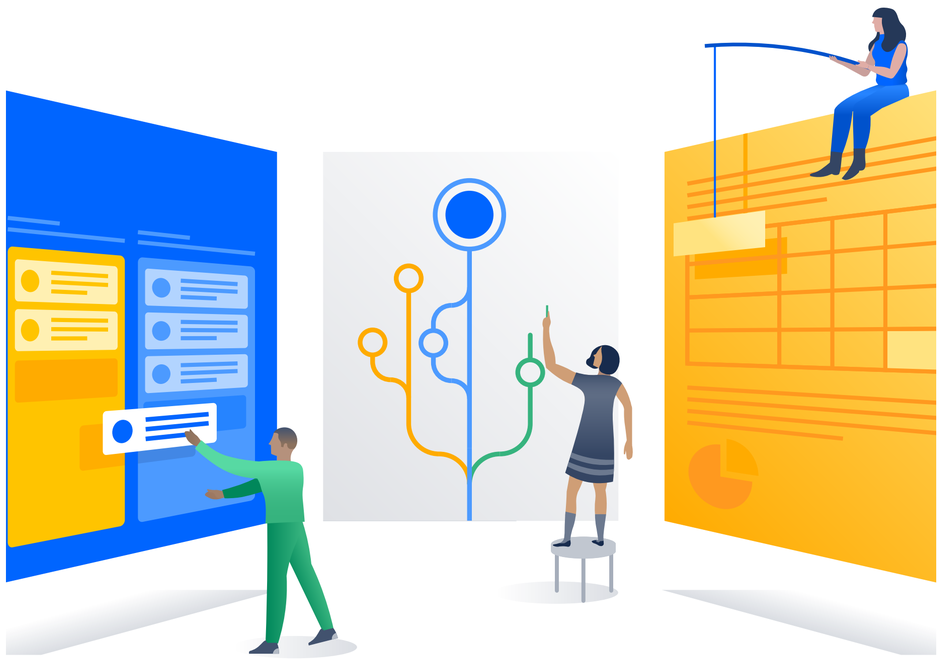
Lightweight

You can run a lot of containers on a single machine.



Continuous Integration

Image scanning, image signing, version tagging



Automated deploys

How do we deploy our container-based application to the cloud?

Agenda

Why containers?

Get your containers online

i2kit

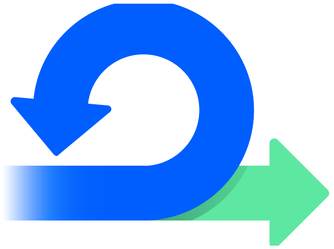
Good practices

Replicable

Test, stage and prod are the same

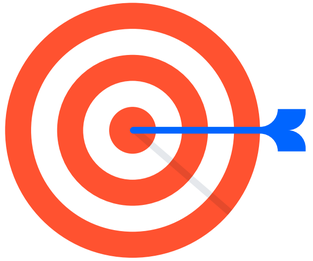
Standard

All services are managed the same way



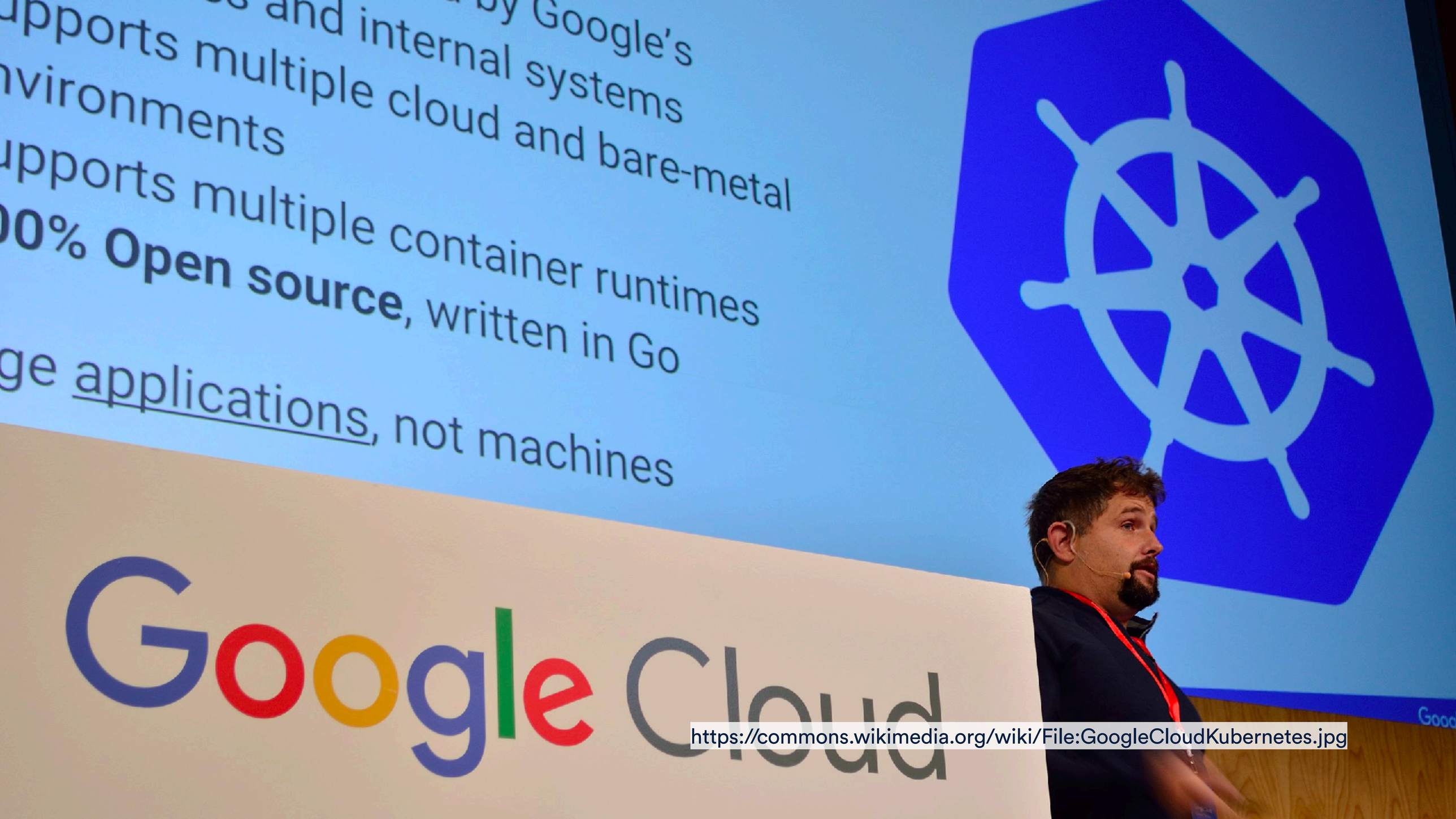
Available

Always online. Like water.



Traceable

Health and status at a glance



https://commons.wikimedia.org/wiki/File:GoogleCloudKubernetes.jpg

Kubernetes is not a deployment tool for containers, it's a different development paradigm

Kubernetes' main strength is optimizing your infrastructure utilization, but at a cost


LAGO
OLIVES

But Kubernetes is not the only way. There are ways better suited for most use cases

For most use cases, we can use cloud native features to manage and deploy our applications

Cloud providers are very robust orchestrators, well fit for containerized applications

Cloud Containerized Application

VMs are the prime deployment unit for cloud, it integrates well and is secure

A container in a VM gives us the best of all worlds: replicable, standard, available and traceable

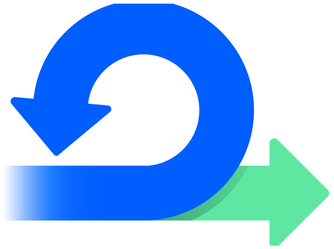
Good practices

Replicable

Test, stage and prod are the same

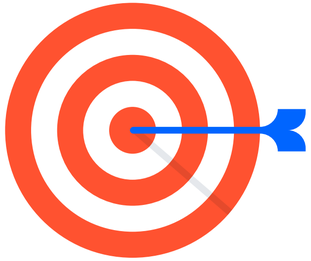
Standard

All services are managed the same way



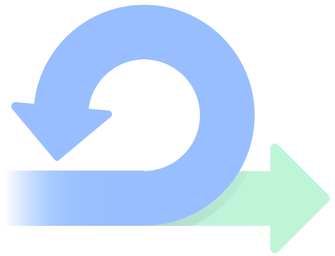
Available

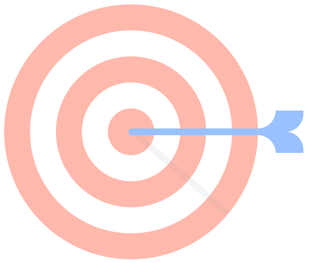
Always online. Like water



Traceable

Health and status at a glance



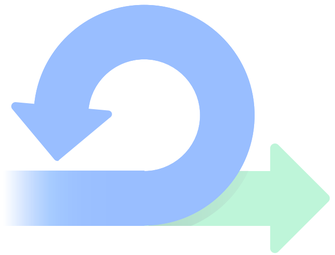


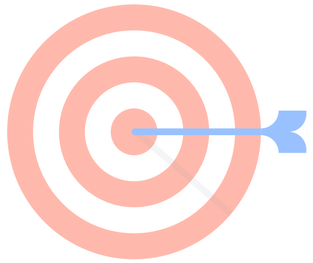
Replicable

Test, staging and production environments are the same, just different scale.

Containers are a great fit for this, they solve dependency and replicability for you.

Use tags and registries to make the same container available to different environments.





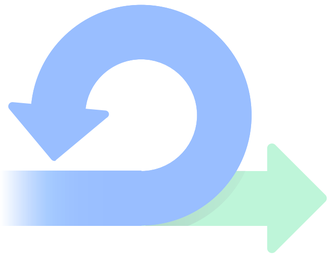
Replicable

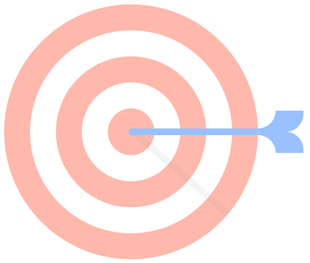
Once deployed, the application never changes.

Virtual machines are great artifacts for this.

Linux kit allows us to create minimalistic VMs. Faster to deploy, and smaller footprint.

Cloud formation (and similar technologies) to automate deploy and upgrade.



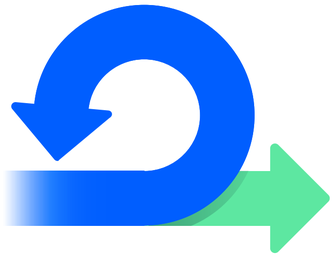


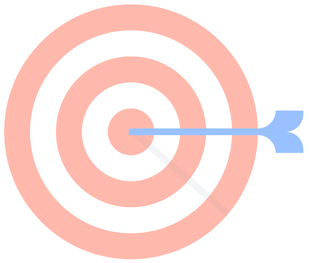
Standard

All services are deployed the same way, independently of the environment

Standardize on a service stack (cloudformation, route53, ec2, cloud watch, containers, registries …) so everybody is trained on the same tools

All teams benefit from playbooks and operational excellence





Available

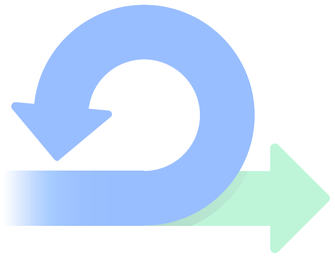
Application should always be available.

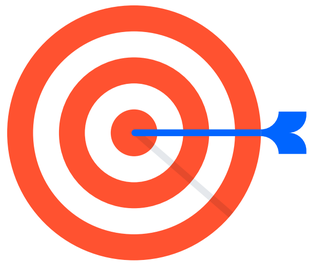
Even during upgrades or peak times.

Elastic load balancers and auto scaling groups can help with peaks and with failover.

CloudFormation and autoscaling groups for rolling upgrades.

Route 53 for blue/green deployments.





Traceable

Know the state of your application at a glance.

The container can be traced to the commit that was used to build it

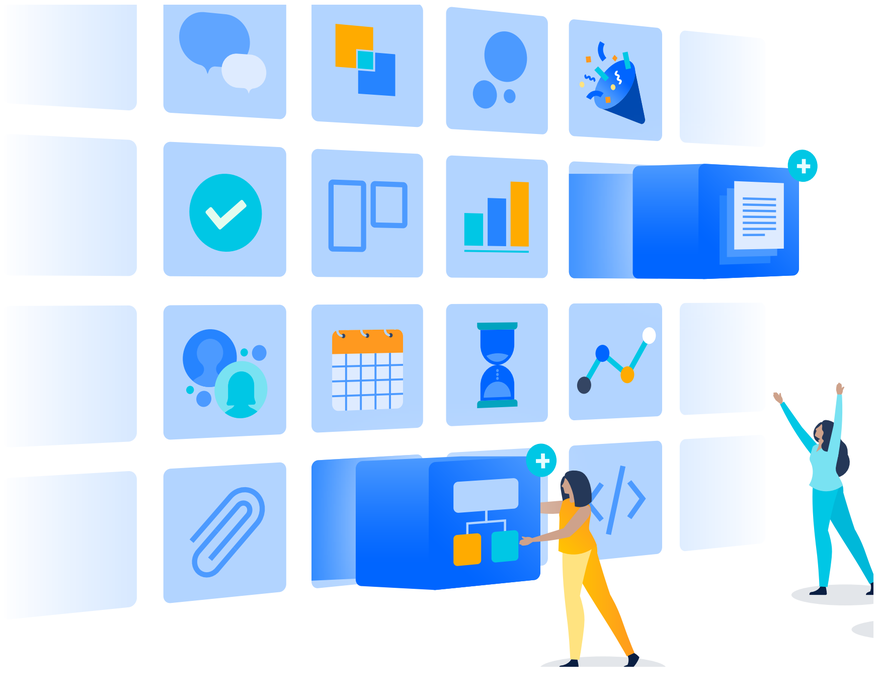
Containers allows you to standardize log management. Just use Amazon CloudWatch Logs logging driver.

CloudWatch can be used to collect CPU, Memory and similar VM-level metrics.

In these slides we use AWS as an example, but the same can be achieved with any other cloud provider

VMs can be big and slow, but there's a lot of research into solving these problems

My VM is lighter (and safer) than your container: https://dl.acm.org/citation.cfm?id=3132763



But how do we do it?

CloudFormation, Terraform, Ansible and others can give your this.

Every team spends countless hours writing and maintaining scripts to deploy and upgrade their own applications

Time spent on this is time not spent on our core competencies.

Agenda

Why containers?

Get your containers online

i2kit

WE WANT TO BENEFIT FROM THE POWER OF THE CLOUD

WE WANT A FORMAT WE'RE ALREADY FAMILIAR WITH

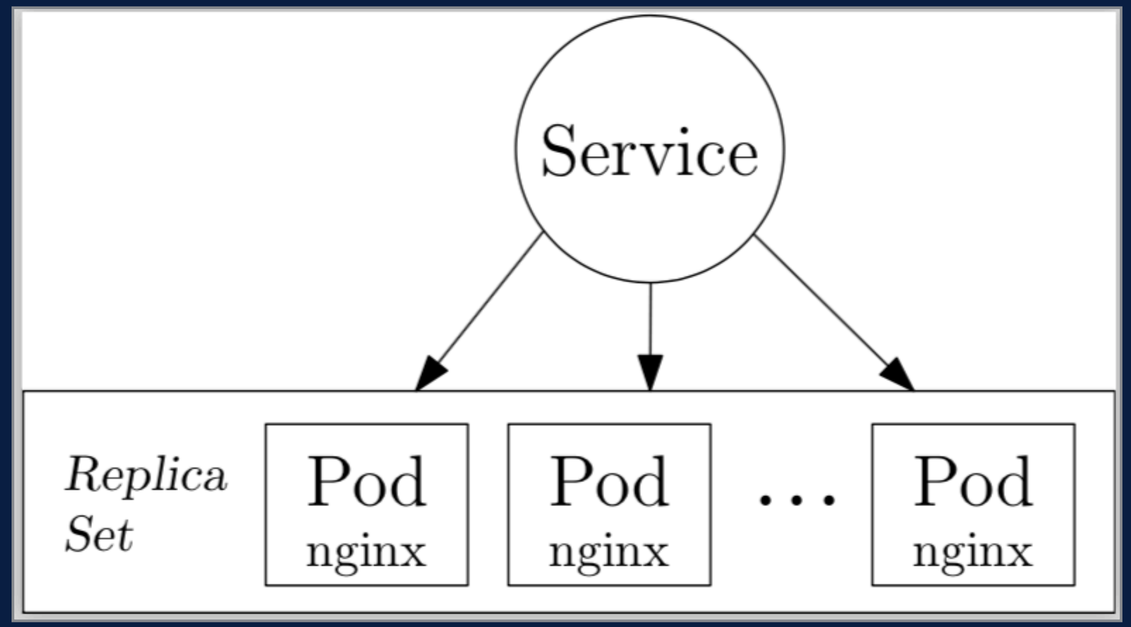
WE DON’T WANT TO MANAGE A CLUSTER

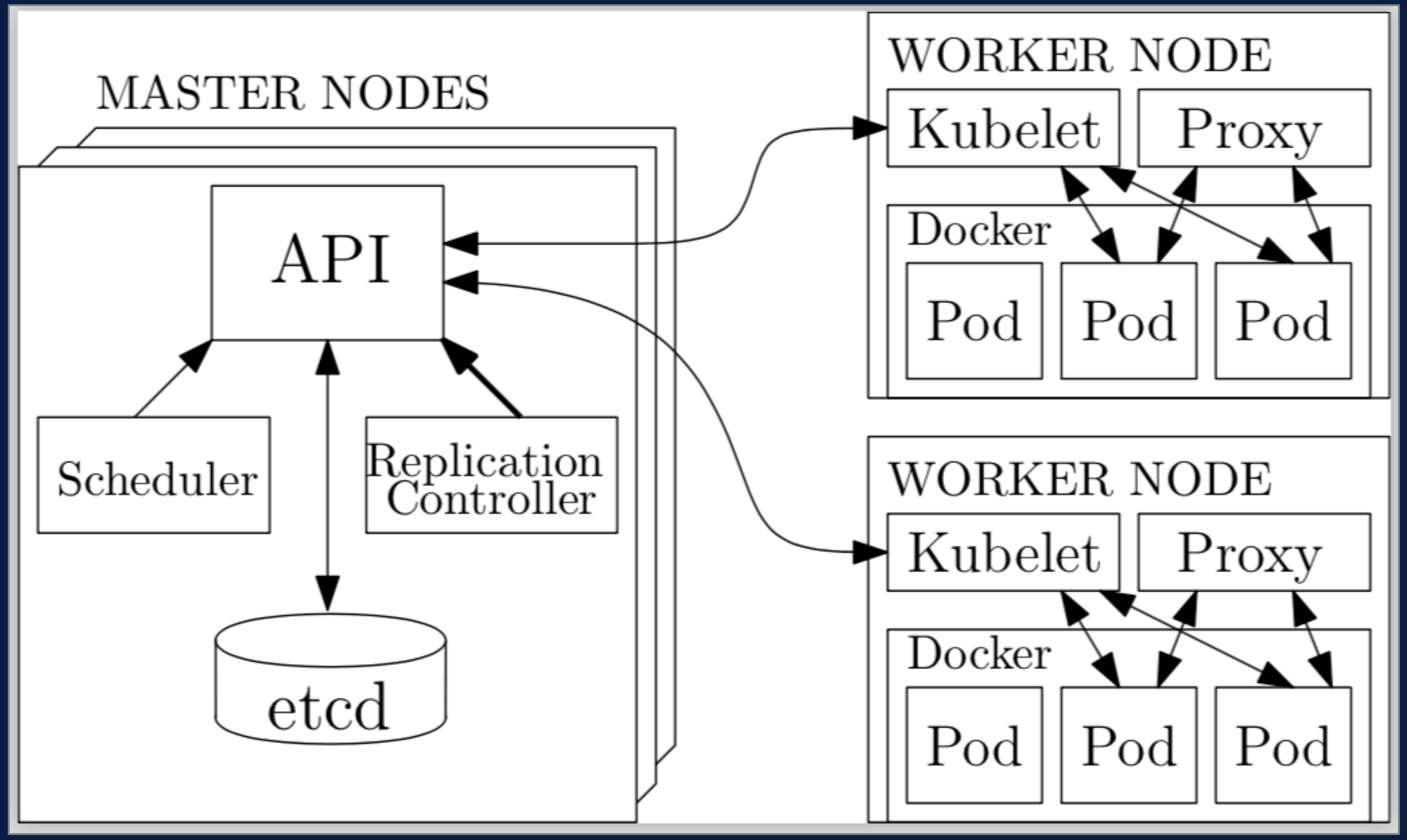
WE TRUST THE VM AS A SECURITY BOUNDARY

MANIFEST

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
    name: webserver
spec:
    selector:
        matchLabels:
            app: webserver
    replicas: 3
    template:
        metadata:
            labels:
                app: webserver
        spec:
            containers:
                - name: webserver
                  image: nginx:1.12-alpine
                  ports:
                    - containerPort: 80
                  env:
                    - name: DOMAIN
                      value: http://riberaproject.com
```

DEPLOYMENT

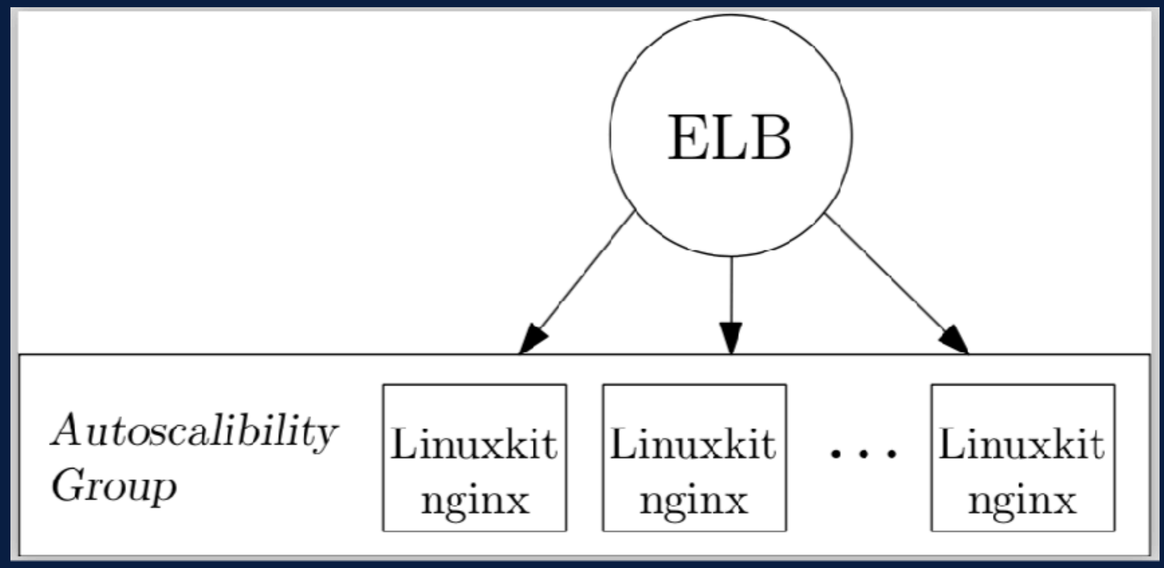



MASTER NODES
API
Scheduler
Replication Controller
etcd
WORKER NODE
Kubelet
Proxy
Docker
Pod
Pod
Pod
WORKER NODE
Kubelet
Proxy
Docker
Pod
Pod
Pod

MANIFEST

```
name: webserver
replicas: 3
containers:
  nginx:
    image: nginx:1.12-alpine
    ports:
      - http:80:http:8000
    environment:
      - DOMAIN=http://riberaproject.com
```

DEPLOYMENT



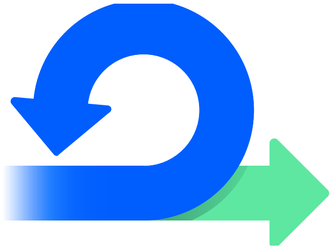
Good practices

Replicable

Same containers everywhere

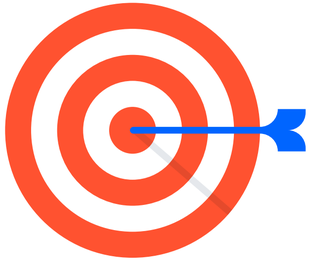
Standard

All services are deployed using manifests



Available

ELBs, ASGs and Route53



Traceable

CloudWatch logs and metrics

https://github.com/pchico83/i2kit

Thank you!

RAMIRO BERRELLEZA | ARCHITECT | @RBERRELLEZA

PABLO CHICO DE GUZMÁN | SENIOR SOFTWARE ENGINEER | @CHICO_DE_GUZMAN