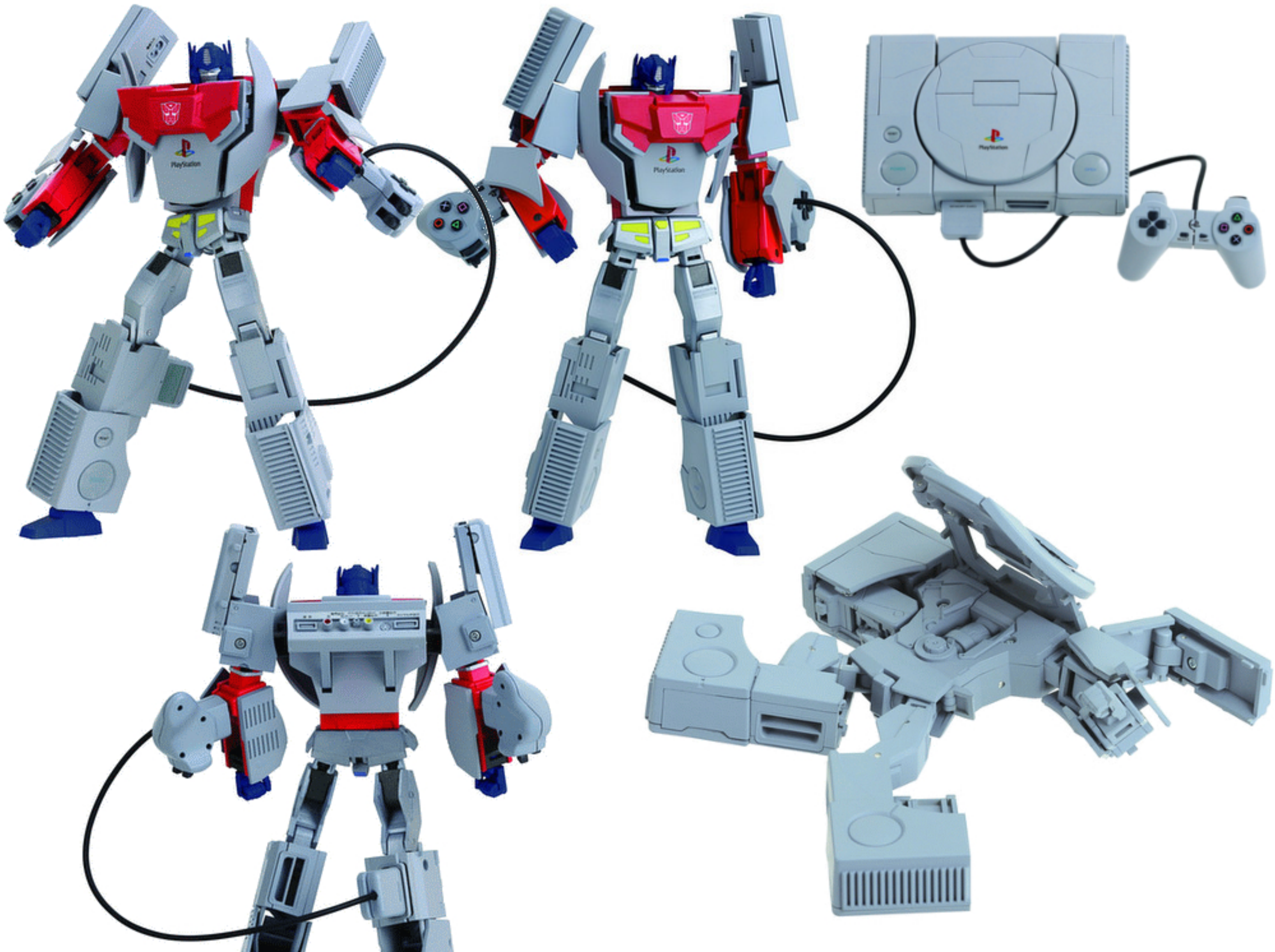
METEOR

+


PlayStation
PlayStation
PlayStation

BTW, I'm @Rahul!

Running US office of @Q42

Explorers & creative technologists

- 60 engineers in Mountain View & NL
- Early start on App Engine in '08
- Early start with Meteor in '12
- Both now core technologies for us

Jumpstarts

- 1 week project with your startup
- Start Monday, end Friday
- Result is working code thanks in part to Meteor
- More at q42.com

Why Meteor?

- Dramatically faster iteration
- Client & server - fully functional demo
- Everyone can write Javascript

Why Container Engine?

- All the features of Cloud Platform
- Declarative configuration in JSON
- Kubernetes is open source
- Containers, duh!

Getting started

- Install `gcloud` command line
- Install `preview container`
- cloud.google.com/container-engine/docs/before-you-begin

The following code is open source

github.com/q42/meteor-on-gke

Demo

http://130.211.60.131/

What we need

1. Meteor Docker image
2. Script to set everything up
3. JSON configuration
4. Example Meteor app

Step 1: Meteor Docker image

- Add the following Dockerfile to your Meteor app
  - `FROM chees/meteor-kubernetes`
  - `ENV ROOT_URL {{your_hostname}}`
- `docker build`
- `docker push`

Step 2: Set everything up

```
gcloud preview container clusters create meteor

gcloud preview container pods create
    --config-file mongo-pod.json

gcloud preview container services create
    --config-file mongo-service.json

gcloud preview container replicationcontrollers create
    --config-file meteor-controller.json
gcloud preview container services create
    --config-file meteor-service.json

gcloud compute firewall-rules create meteor-80
    --allow=tcp:80
    --target-tags k8s-meteor-node
```

Step 3: JSON configuration

1. Setting up a pod

```
{
    "id": "mongo",
    "kind": "Pod",
    "desiredState": {
        ...
    },
    "labels": {
        ...
    }
}
```

Setting up a pod (2)

```
"containers": [{
    "name": "mongo",
    "image": "mongo",
    "cpu": 1000,
    "ports": [{ "name": "mongo", "containerPort": 27017 }],
    "volumeMounts": [{
        "mountPath": "/data/db",
        "name": "mongo-disk"
    }]
}]
```

Setting up a pod (3)

```
"volumes": [{
    "name": "mongo-disk",
    "source": {
        "persistentDisk": {
            "pdName": "mongo-disk",
            "fsType": "ext4"
        }
    }
}]
```

2. Setting up a replication controller

```
{
    "id": "meteor-controller",
    "kind": "ReplicationController",
    "apiVersion": "v1beta1",
    "desiredState": {
        ...
    },
    "labels": {"name": "meteor"}
}
```

Setting up a replication controller (2)

```
"replicas": 3,
"replicaSelector": {"name": "meteor"},
"podTemplate": {
    "desiredState": {
        "manifest": {
            "version": "v1beta1",
            "id": "meteor-controller",
            "containers": [{
                "name": "meteor",
                "image": "chees/meteor-gke-example",
                "cpu": 1000,
                "memory": 500000000,
                "ports": [{"name": "http-server", "containerPort": 8080, "hostPort": 80}]
            }]
        }
    },
    "labels": { "name": "meteor" }
}
```

3. Setting up the Mongo service

```
{
    "id": "mongo",
    "kind": "Service",
    "apiVersion": "v1beta1",
    "port": 27017,
    "containerPort": "mongo",
    "selector": {
        "name": "mongo", "role": "mongo"
    },
    "labels": {
        "name": "mongo"
    }
}
```

4. Setting up the Meteor service

```
{
    "apiVersion": "v1beta1",
    "kind": "Service",
    "id": "meteor",
    "port": 80,
    "containerPort": "http-server",
    "selector": { "name": "meteor" },
    "createExternalLoadBalancer": true,
    "sessionAffinity": "ClientIP"
}
```

Step 4. Example Meteor app

registry.hub.docker.com/u/chees/meteor-gke-example

Just change the number of replicas to scale

```
"replicas": 5,

"replicaSelector": {"name": "meteor"},
"podTemplate": {
    ...
}
```

What about updating your app?

```
gcloud preview container replicationcontrollers delete meteor-controller

gcloud preview container replicationcontrollers create
    --config-file meteor-controller.json

# Rolling update
OLD_PODS=`gcloud preview container pods list | grep name=meteor | cut -f1 -d ' '`
while read -r POD; do
    gcloud preview container pods delete $POD
    # You might want to do the rolling update slower in practice:
    sleep 30
done <<< "$OLD_PODS"
```

Takeaways

1. Expressing scaling declaratively is awesome
2. Not thinking about the underlying hardware is awesome
3. Viewing the cloud as a single CPU is awesome

This is just the start. Now what?

- Extend Meteor command line?
  - `meteor deploy gke?`
- Auto-scaling?
- Fix MongoDB bottleneck?
- ??? What do you need?

Get involved!

github.com/q42/meteor-on-gke

More reading

- meteor.com
- github.com/GoogleCloudPlatform/Kubernetes
- cloud.google.com/container-engine

Thanks to Christiaan Hees

Who did all the work. I just talked about it. :)

@christiaanhees

Btw, here's something from Google:

$500 in Google Cloud Platform credit!

- Go to cloud.google.com/startercredit
- Click Apply Now
- Complete the form with code: `meteor-org`

Thanks for coming :)

Nestlé®
GOOBERNETES
Nestlé® Milk Chocolate • Fresh Roasted Peanuts
Chocolate con leche Nestlé® Cacahuetes (maní) tostados y frescos
5 GRAMS OF PROTEIN PER SERVING
see nutrition information for fat and saturated fat content
NET WT / PESO NETO 1.38 OZ (39.1g)
TEAR HERE

- Next week: "Introduction to Meteor"
  - Jan 21st, 6pm, same place
  - meetup.com/javascript-9
- Have a venue? Get in touch!
  @q42 or @rahul