+ Auto-scale applications on Cloud Foundry

Ying Liu
Yang Qi

The team

- A community project hosted in

https://github.com/cloudfoundry-incubator/app-autoscaler

- Major contributors
  - IBM: Bo Yang, Yang Qi, Ying Liu
  - SAP: Pradyut , Rohit, Tanmoy
  - Fujitsu: Hiroyuki Kaneko, Ghaih

Manage Application Capacity with Auto-Scaling

Without Auto-Scale

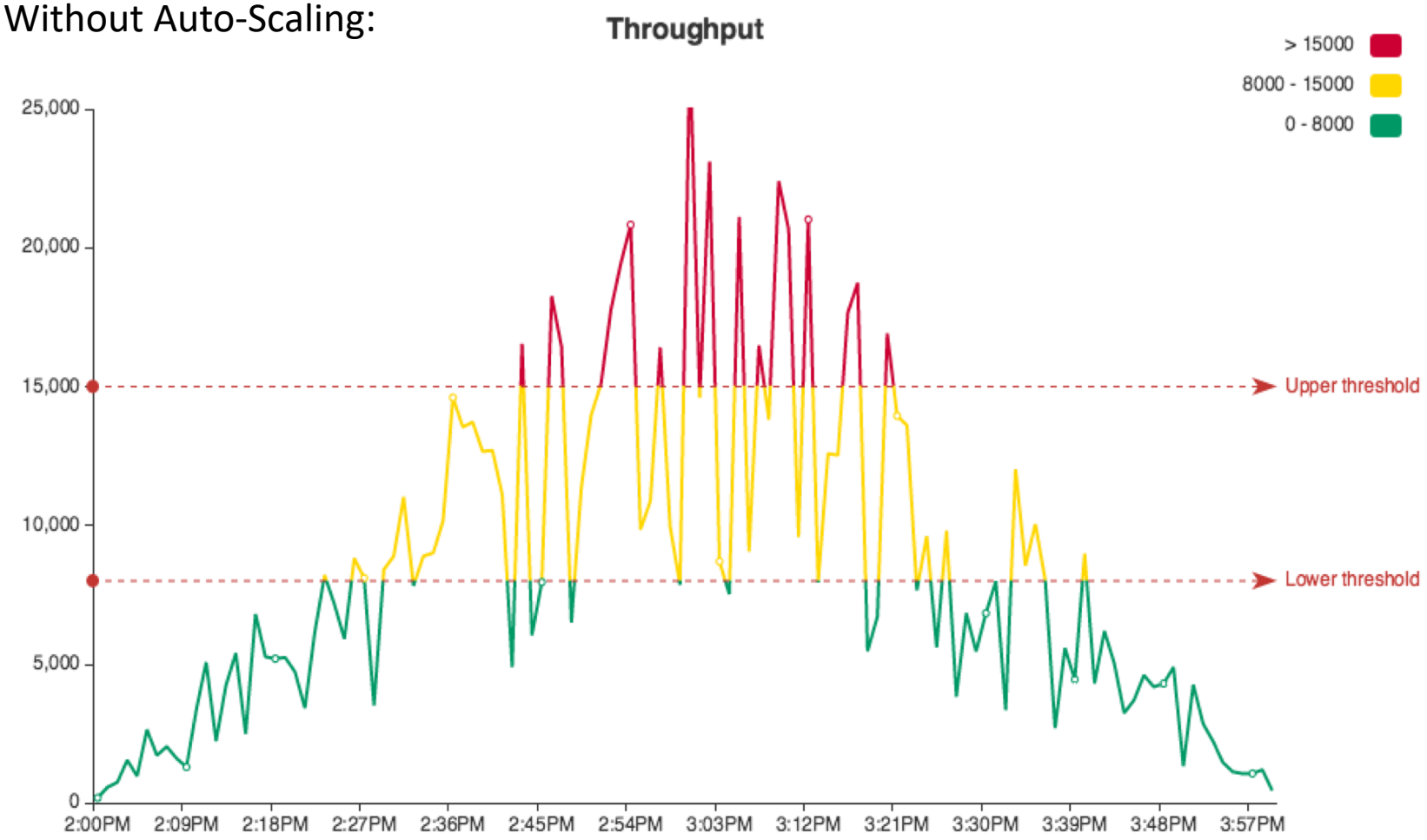
- Monitor application
- Change instance manually
  - cf scale MYAPP –i 3
  - cf scale MYAPP –i 1
- Keep monitoring …

With Auto-Scale

- Apply scaling policy

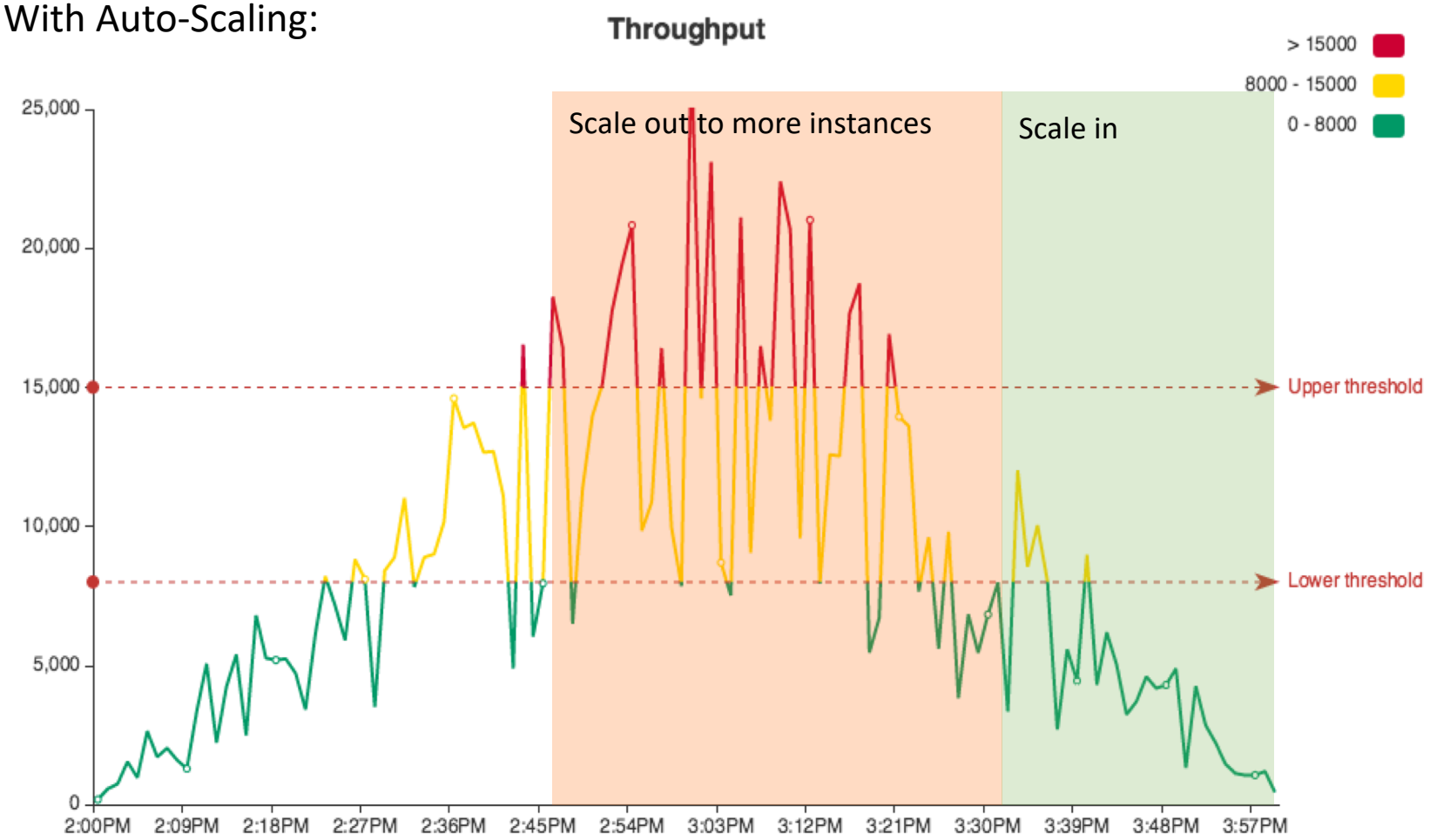
Dynamic Scaling

Without Auto-Scaling:



Dynamic Scaling

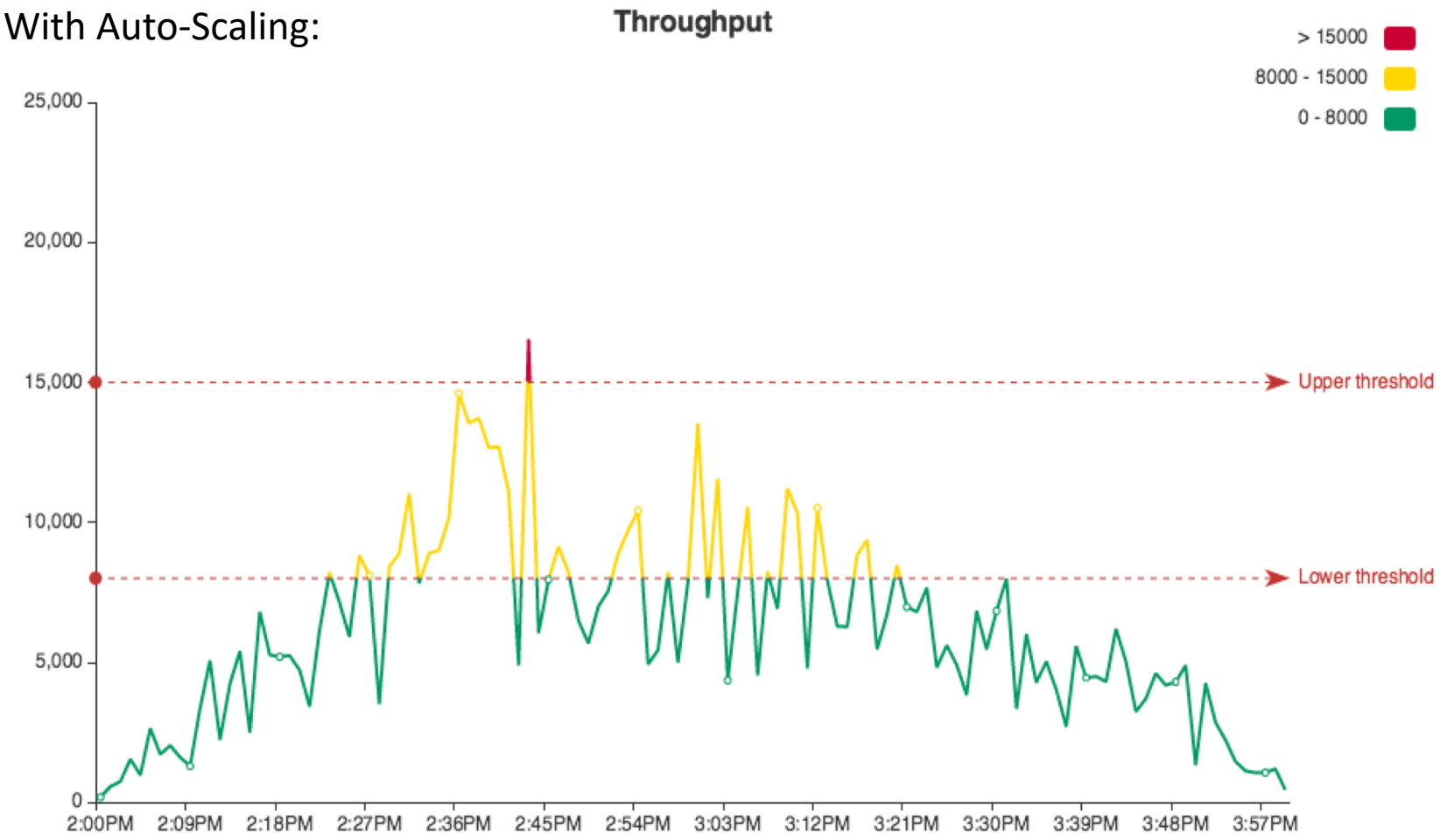
With Auto-Scaling:




```
{
  "instance_min_count": 1,
  "instance_max_count": 4,
  "scaling_rules": [
  {
    "metric_type": "throughput",
    "breach_duration_secs": 600,
    "threshold": 7000,
    "operator": "<",
    "cool_down_secs": 300,
    "adjustment": "-1"
  },
  {
    "metric_type": "throughput",
    "breach_duration_secs": 600,
    "threshold": 15000,
    "operator": ">=",
    "cool_down_secs": 300,
    "adjustment": "+1"
  }]
}
```

Dynamic Scaling

With Auto-Scaling:



- Memory Used
- Memory Utilization
- Throughput
- Response Time
- Custom metrics (coming soon)

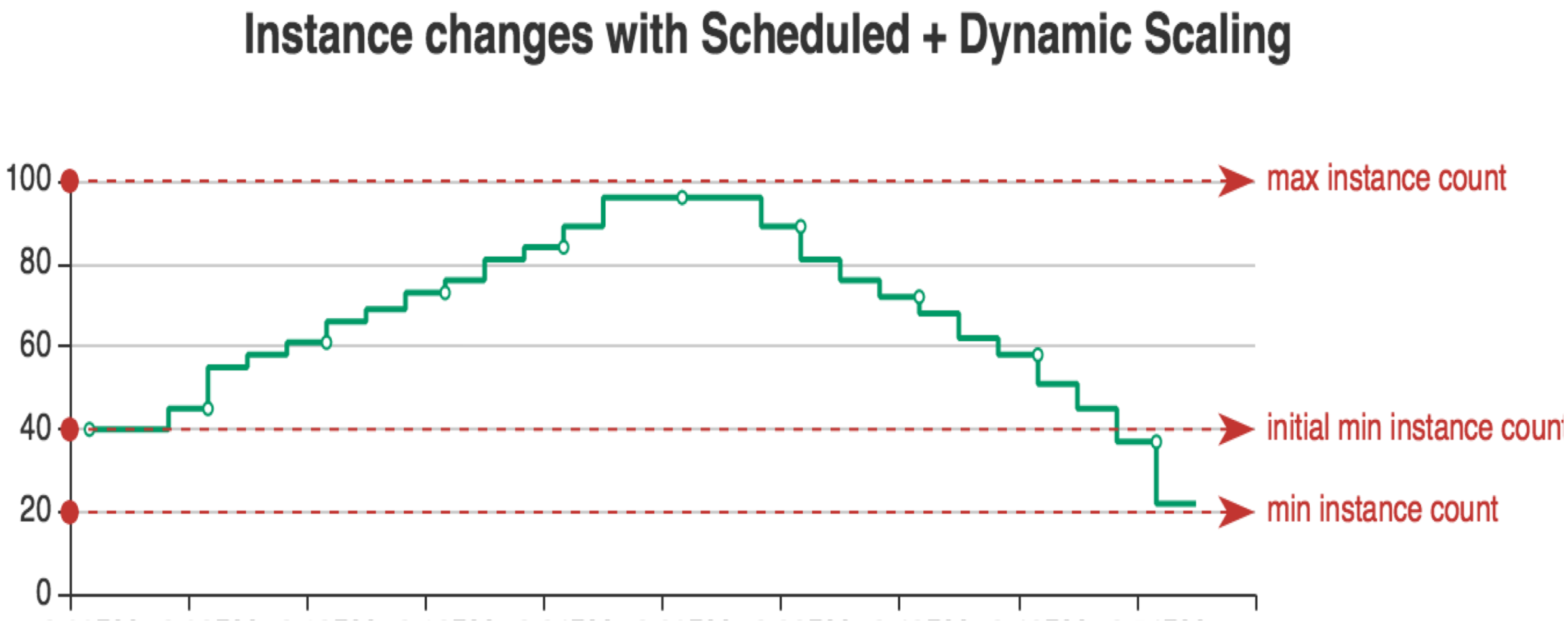
Scheduled Scaling

- Recurring Schedule
  - Day-of-Week ,Day-of-Month
  - Start time ~ End time

- Specific Date Schedule
  - Start date / time ~ End date / time

```
{
  "instance_min_count": 1,
  "instance_max_count": 4,
  "schedules":
  {
    "timezone": "Asia/Shanghai",
    "recurring_schedule": [
    {
      "start_time": "10:00",
      "end_time": "18:00",
      "days_of_week": [1,2,3],
      "instance_min_count": 1,
      "instance_max_count": 10,
      "initial_min_instance_count": 5
    }],
    "specific_date": [
    {
      "start_date_time": "2018-08-08T14:00",
      "end_date_time": "2018-08-08T14:59",
      "instance_min_count": 1,
      "instance_max_count": 4,
      "initial_min_instance_count": 2
    }]
  }
}
```

Dynamic + Scheduled Scaling

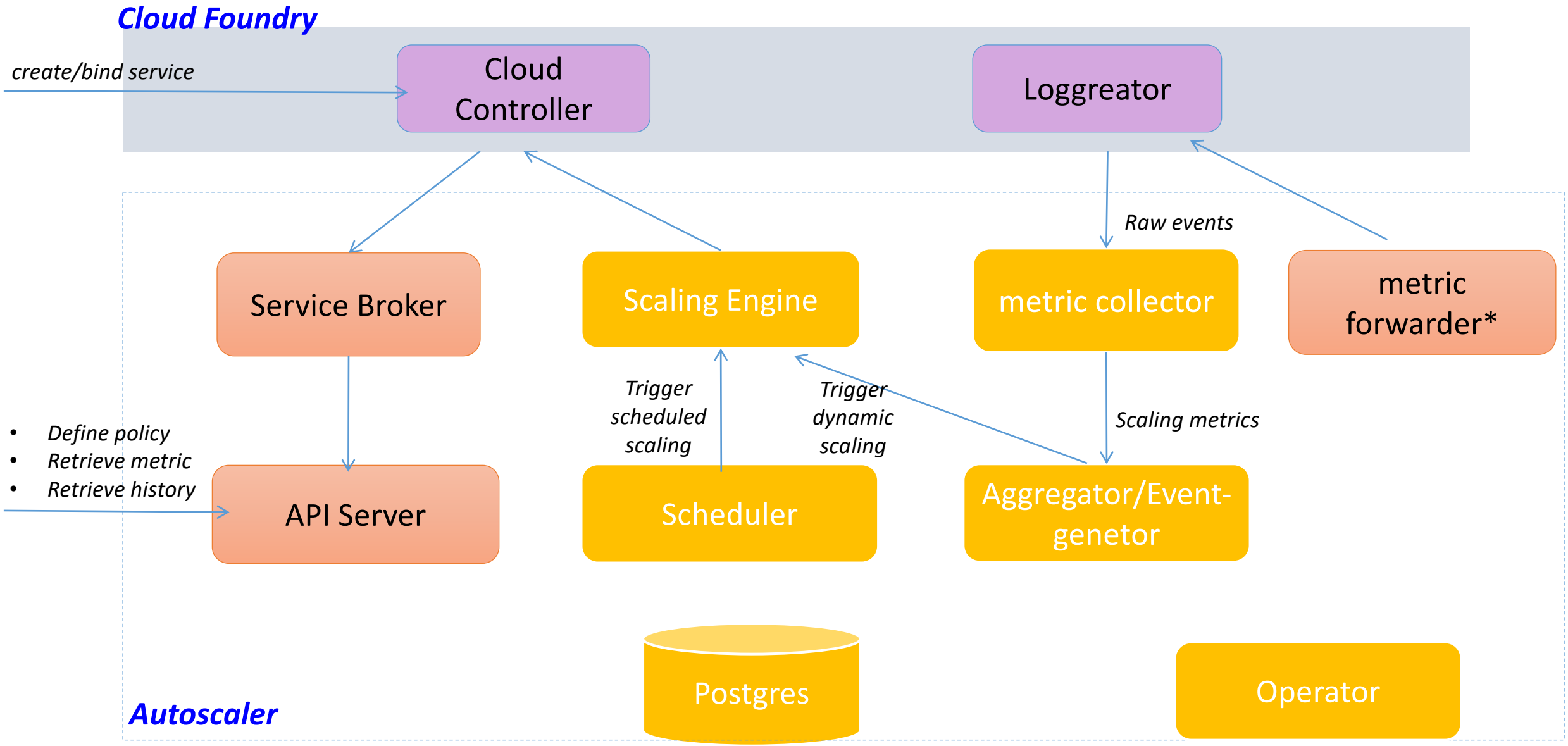


Operate with Command Line

- Install app-autoscaler-plugin
  - *cf install-plugin -r CF-Community "app-autoscaler-plugin"*
- Attach policy directly with bind-service
  - *cf bind-service demoapp demoservice -c policy.json*
- Attach policy with “app-autoscaler-plugin”
  - *cf asa https://autoscaler.bosh-lite.com*
  - *cf aasp demoapp test.json*
- More operations with “app-autoscaler-plugin”
  - Retrieve policy : cf asp demoapp
  - Retrieve metrics: cf asm demoapp memoryutil
  - Retrieve history: cf ash demoapp

Deep Dive to AutoScaler

Architecture Diagram



Deploy AutoScaler

- Prerequisite :
  - Install Cloud Foundry
  - Download AutoScaler bosh release:

    git clone https://github.com/cloudfoundry-incubator/app-autoscaler-release
- Deploy with cf-deployment
  - bosh create-release && bosh upload-release
  - bosh -e YOUR_ENV -d app-autoscaler deploy templates/app-autoscaler-deployment.yml --vars-store=bosh-lite/deployments/vars/autoscaler-deployment-vars.yml -v system_domain=bosh-lite.com -v cf_admin_password=<cf admin password>

Offer Auto-Scaling with Different Approach

Offer as a service

- Auto-Scaler Provider:
  - Deploy autoscaler release…
  - Create service offering…
- End-user:
  - cf create-service …
  - cf bind-service <app> <service instance> -c <policy file>

Offer as a build-in experience

- Auto-Scaler Provider:
  - Deploy autoscaler release…
- End-user:
  - cf autoscaling-api …
  - cf attach-autoscaling-policy <app> <policy file>

Deploy AutoScaler on Kubernetes

```
NAME                    READY   STATUS    RESTARTS   AGE
api-0                   1/1     Running   0          4h
autoscaler-actors-0     1/1     Running   0          4h
autoscaler-api-0        1/1     Running   0          4h
autoscaler-metrics-0    1/1     Running   0          4h
autoscaler-postgres-0   1/1     Running   0          4h
blobstore-0             1/1     Running   0          4h
cc-clock-0              1/1     Running   0          4h
cc-uploader-0           1/1     Running   0          4h
cc-worker-0             1/1     Running   0          4h
cf-usb-0                1/1     Running   0          4h
diego-access-0          1/1     Running   0          4h
diego-api-0             1/1     Running   0          4h
diego-brain-0           1/1     Running   0          4h
diego-cell-0            1/1     Running   0          4h
diego-locket-0          1/1     Running   0          4h
doppler-0               1/1     Running   0          4h
loggregator-0           1/1     Running   0          4h
mysql-0                 1/1     Running   0          4h
mysql-proxy-0           1/1     Running   0          4h
nats-0                  1/1     Running   0          4h
nfs-broker-0            1/1     Running   0          4h
postgres-0              1/1     Running   0          4h
router-0                1/1     Running   0          4h
routing-api-0           1/1     Running   0          4h
syslog-adapter-0        1/1     Running   0          4h
syslog-rlp-0            1/1     Running   0          4h
syslog-scheduler-0      1/1     Running   0          4h
tcp-router-0            1/1     Running   0          4h
```

Coming Soon …

- Support more metrics
  - Custom metrics support
  - CPU metrics support
- Performance enhancement
- Loggregator /v2 API
- Multiple cloud support

….

Welcome contributions !

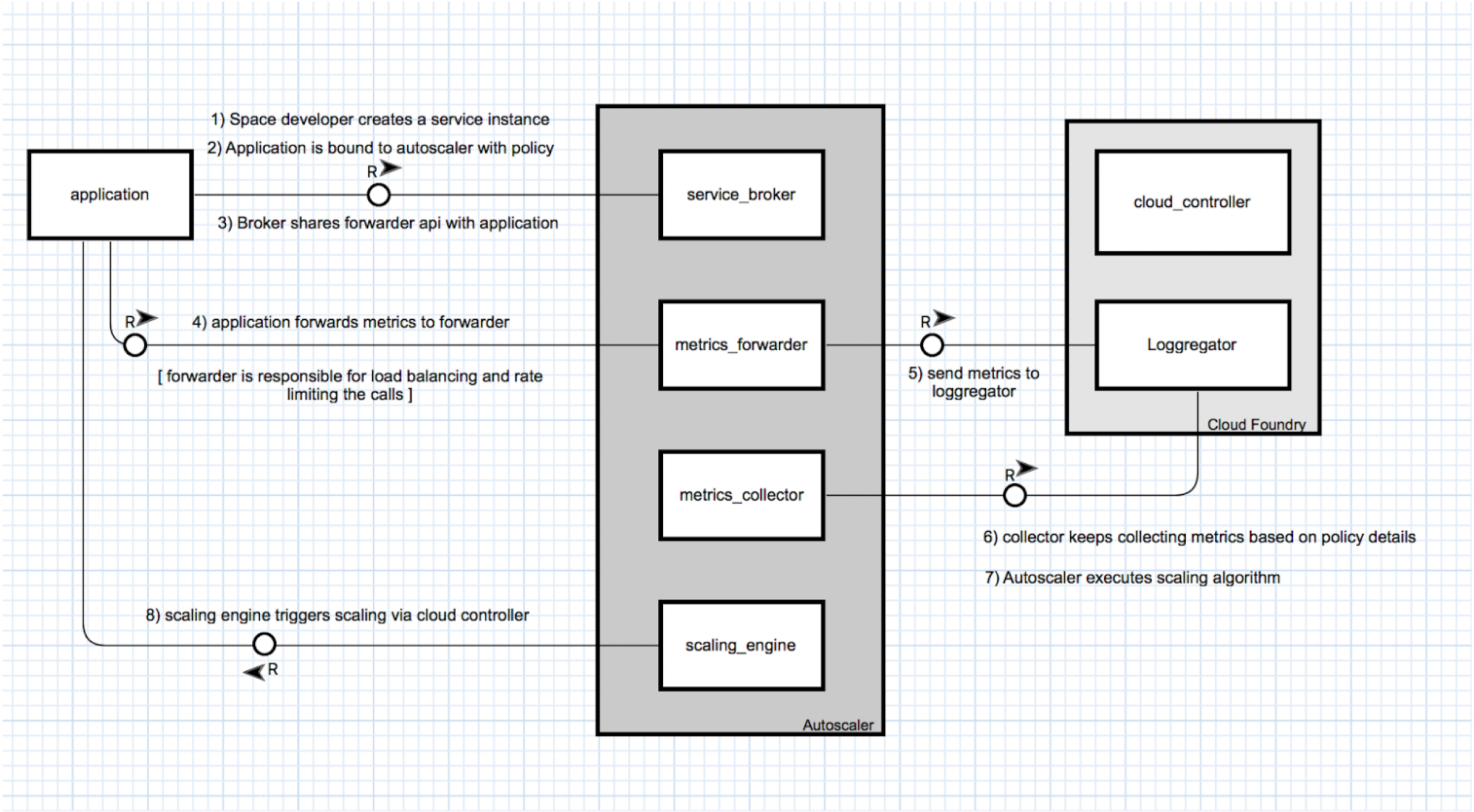
Thank you !

- GIT :
  https://github.com/cloudfoundry-incubator/app-autoscaler
  https://github.com/cloudfoundry-incubator/app-autoscaler-release

- Slack: #autoscaler on cloudfoundry.slack.com
  https://cloudfoundry.slack.com/messages/autoscaler/

Backup

Custom metric support



CPU metrics

- CPU entitlement proposal in Garden container
  - Containers should have a ‘CPU entitlement’ (still proportional to requested memory limit, for now) which is the amount of CPU a container is entitled to average over time.
  - CPU metrics should be reported relative to the entitlement.
  - See more in https://docs.google.com/document/d/16TvBsZSInjy7zoboSQWRJo30lmxr07tvd40_GzpfD3I/edit#

➔ Auto-scaler will bring CPU metric back as a trust measurement of scaling decision

Operate with Command Line

- cf install-plugin -r CF-Community "app-autoscaler-plugin"

Command	Description
autoscaling-api, asa	Set or view AutoScaler service API endpoint
autoscaling-policy, asp	Retrieve the scaling policy of an applicationa
attach-autoscaling-policy, aasp	Attach a scaling policy to an application
detach-autoscaling-policy, dasp	Detach the scaling policy from an application
autoscaling-metrics, asm	Retrieve the metrics of an application
autoscaling-history, ash	Retrieve the scaling history of an application

Policy Content

```
{
    "instance_min_count": 1,
    "instance_max_count": 4,
    "scaling_rules": [{
    }],
    "schedules": {
        "timezone": "Asia/Shanghai",
        "recurring_schedule": [{
        }],
        "specific_date": [{
        }]
    }
}
```

metric_type	breach_duration_secs	threshold	operator	adjustment	cool_down_secs
Throughput	600	8000	<	-1	300
Throughput	600	15000	>	+1	300

start_time	end_time	day_of_week/ day_of_month	instance_min_count	initial_min_ instance_count	instance_max_count
10:00	18:00	1,3,5	3	5	10
18:00	23:00	1,2,3,4,5,6,7	5	8	15

start_date_time	end_date_time	instance_min_count	instance_max_count	initial_min_ instance_count
2017-10-11T10:00	2017-10-13T10:00	5	20	10
2017-12-23T10:00	2017-12-26T10:00	5	20	10

Steps to implement Auto-Scaling:

- Understand the traffic and workload type of the application
- Benchmark the application to understand the performance of the application
- Scale application manually to understand how the application behaves when scaling out (how long it is needed to warm up, what is the impact of existing load and sticky session if it is used)
- Identify the performance bottleneck from the benchmarking result, and decide which metric should be used to dynamically adjust the instance number.
- Define the initial scaling policy and enable Auto-Scaling service
- Simulate the workload and test with Auto-Scaling, to adjust detailed settings of the policy, including thresholds, steps of scaling, statistics window, breach duration and cool-down period
- Define scheduled scaling for peak hours
- Simulate the peak hour workload, and adjust the min/max instance number settings in the scheduled policy
- Apply the refined policy and let it go