ECS & Docker: Secure Async Execution @ coursera

Brennan Saeta

QCon LONDON

coursera

Institutions Log In Sign Up

Take the world's best courses, online.

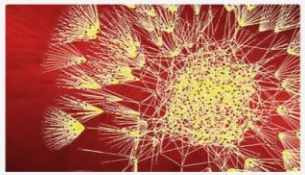
What would you like to learn about?

or browse catalog ›

17,867,235 learners · 1,804 courses · 138 partners

Popular Specializations

Data Science
Johns Hopkins University

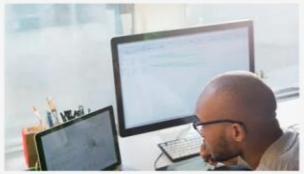


Big Data
University of California, San Di...

Python for Everybody
University of Michigan

Excel to MySQL: Analytic Techniques for Business
Duke University

Business Analytics
University of Pennsylvania



Machine Learning
University of Washington

Introduction to Project Management Principles...
University of California, Irvine

Full Stack Web Development
The Hong Kong University of S...

Most Popular

The Beginnings — 2012

1 million

learners worldwide

4

partners

10

courses

Education at Scale

18 million

learners worldwide

140

partners

1,800

courses

Outline

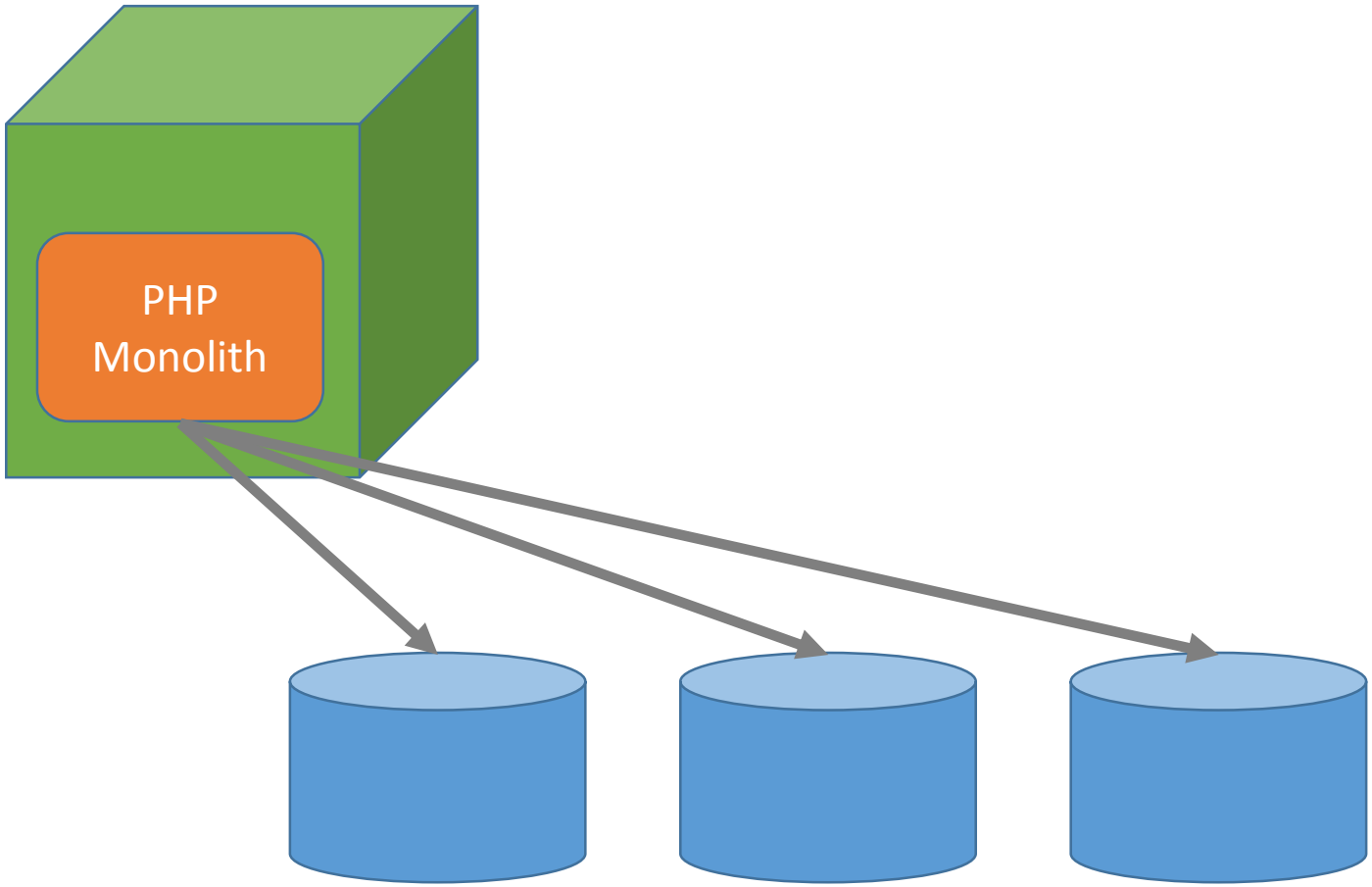
- Evolution of Coursera’s nearline execution systems
- Next-generation execution framework: Iguazú
- Iguazú application deep dive:
  GrID — evaluating programming assignments

Key Takeaways

- What is *nearline* execution, and why it is useful
- Best practices for running containers in production in the cloud
- Hardening techniques for securely operating container infrastructure at scale

A history of nearline execution

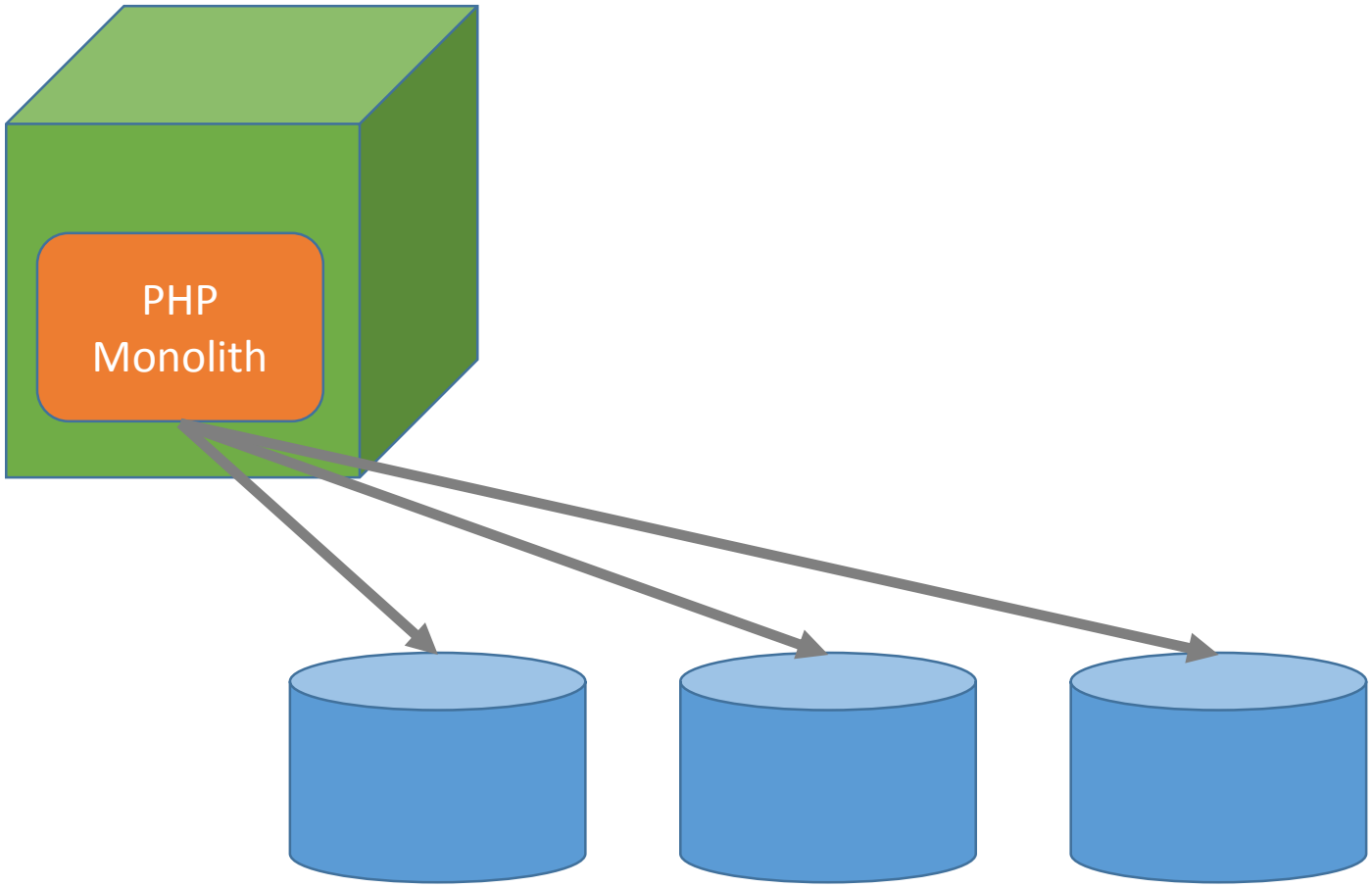
Coursera Architecture (2012)



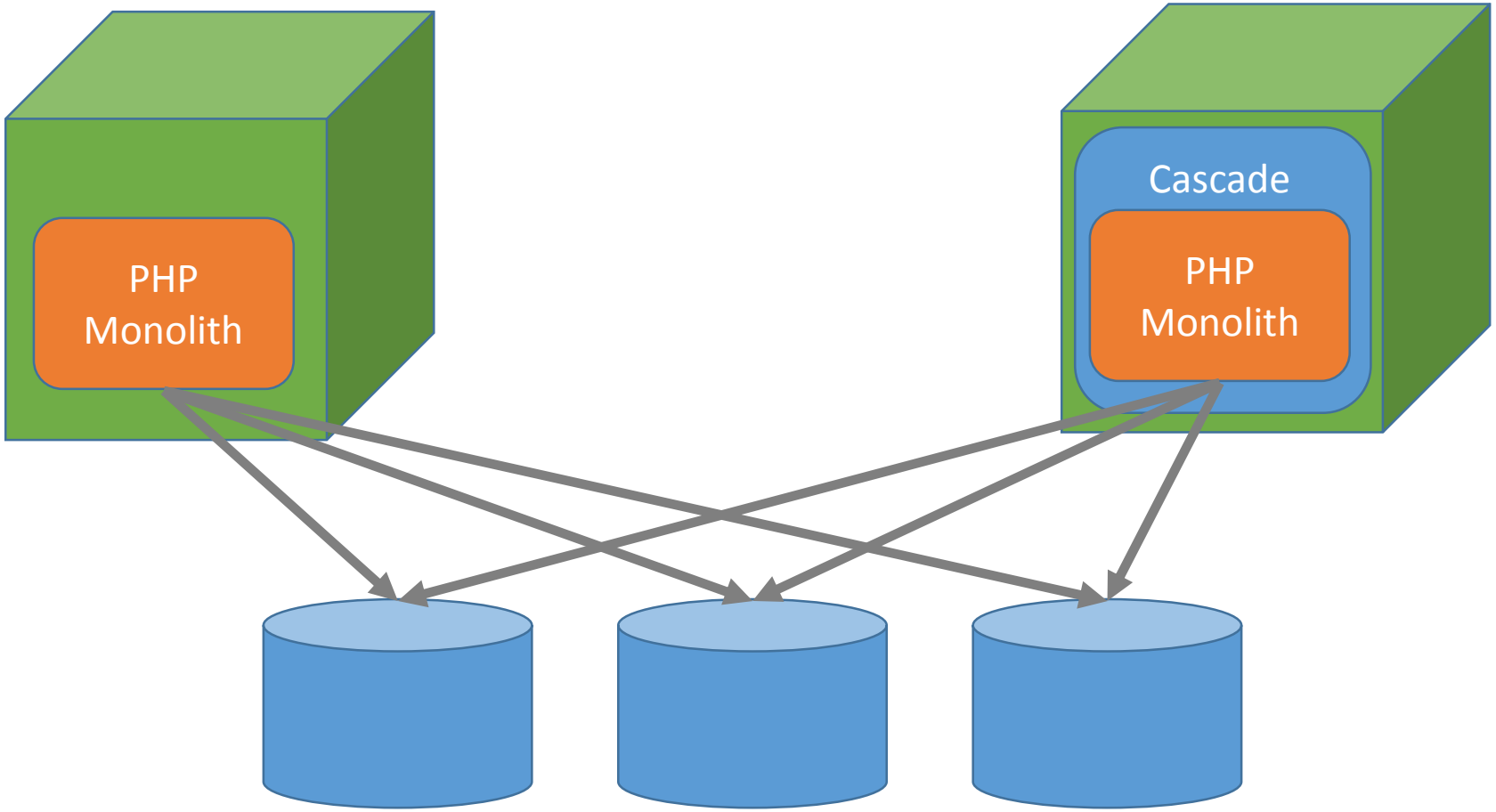
Early days - Requirements

- Video re-encoding for distribution
- Grade computation for 100,000+ learners
- Pedagogical data exports for courses

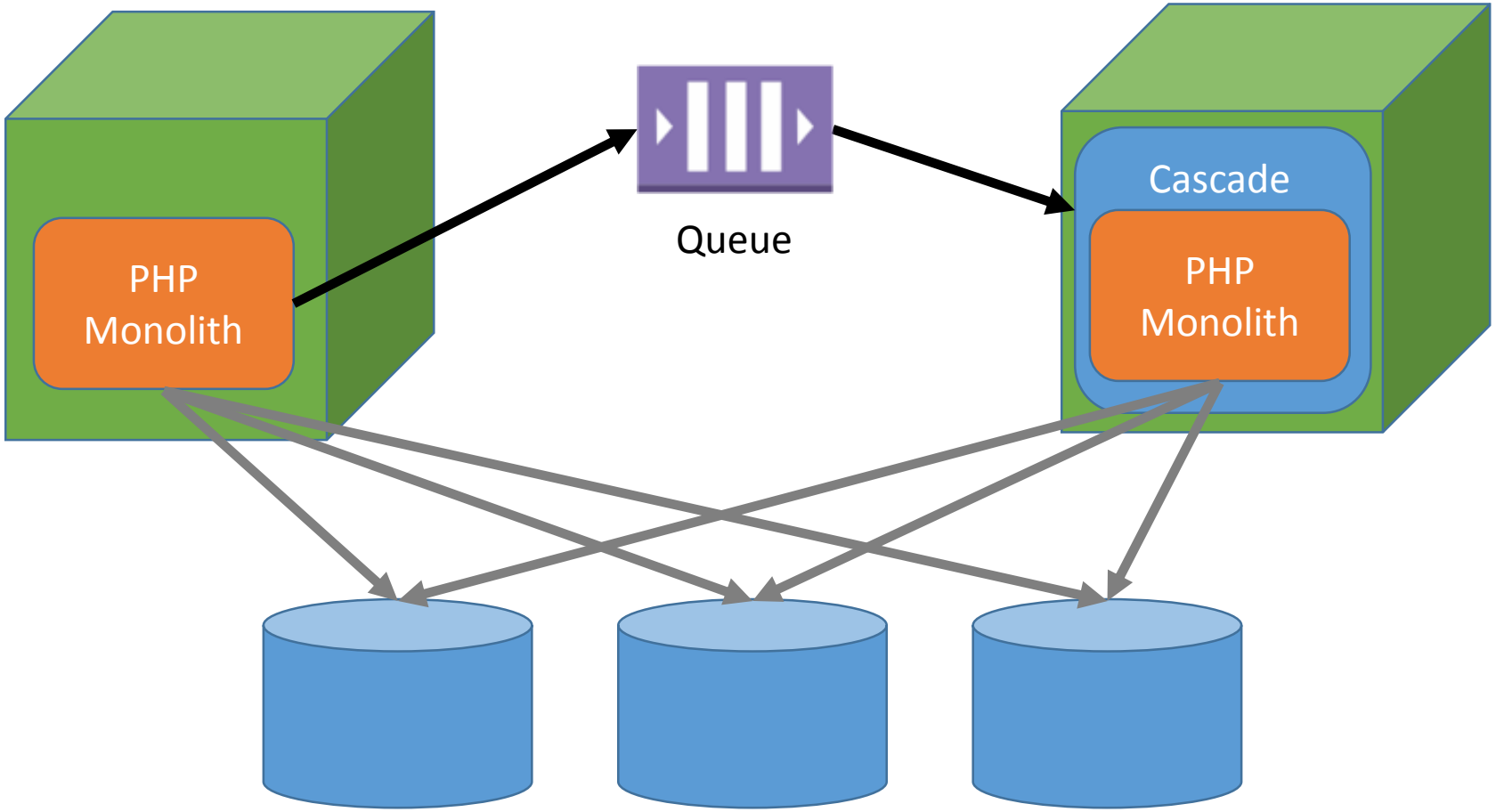
Coursera Architecture (2012)



Cascade Architecture



Cascade Architecture



Upgrading to Scala

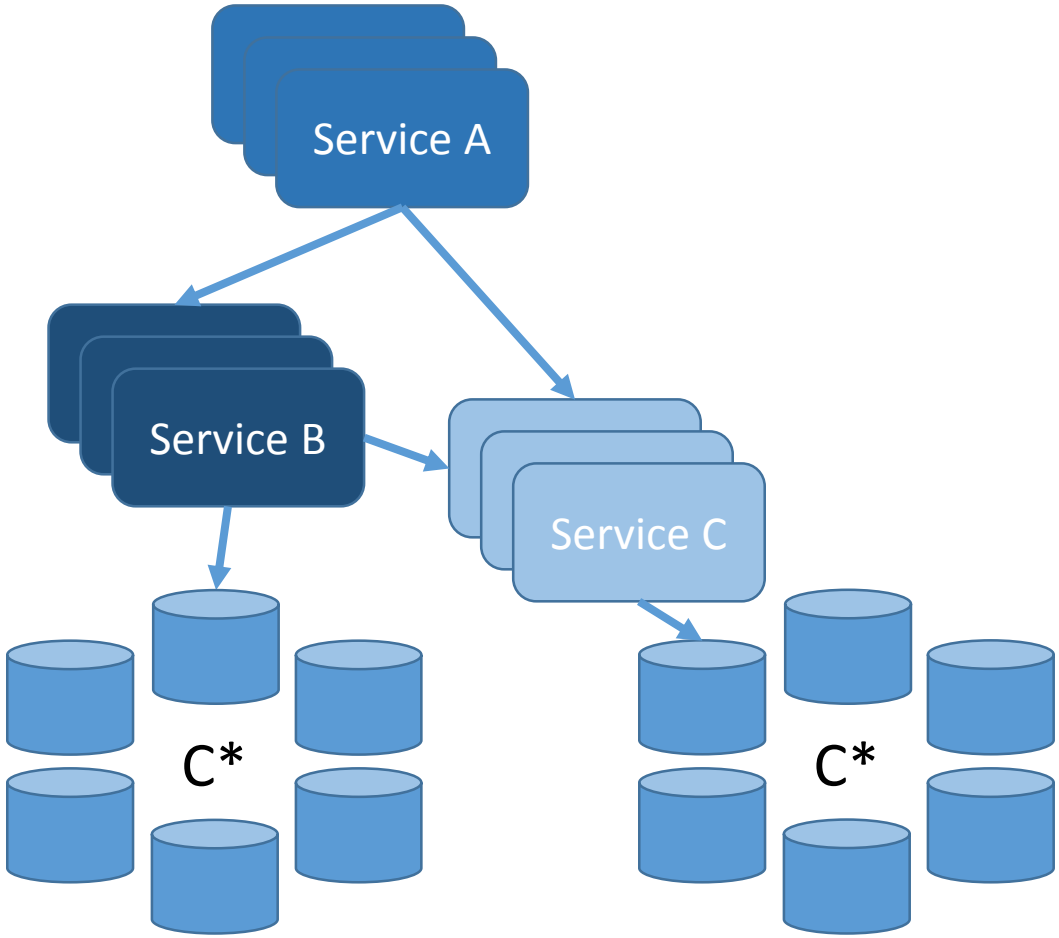
Re-architecting delayed execution for our 2nd generation learning platform.

Upgrading to the JVM

- Leverage mature Scala & JVM ecosystems for code sharing
- JVM much more reliable (no memory leaks)
- New job model: scheduled recurring jobs.
  - Named: Saturn

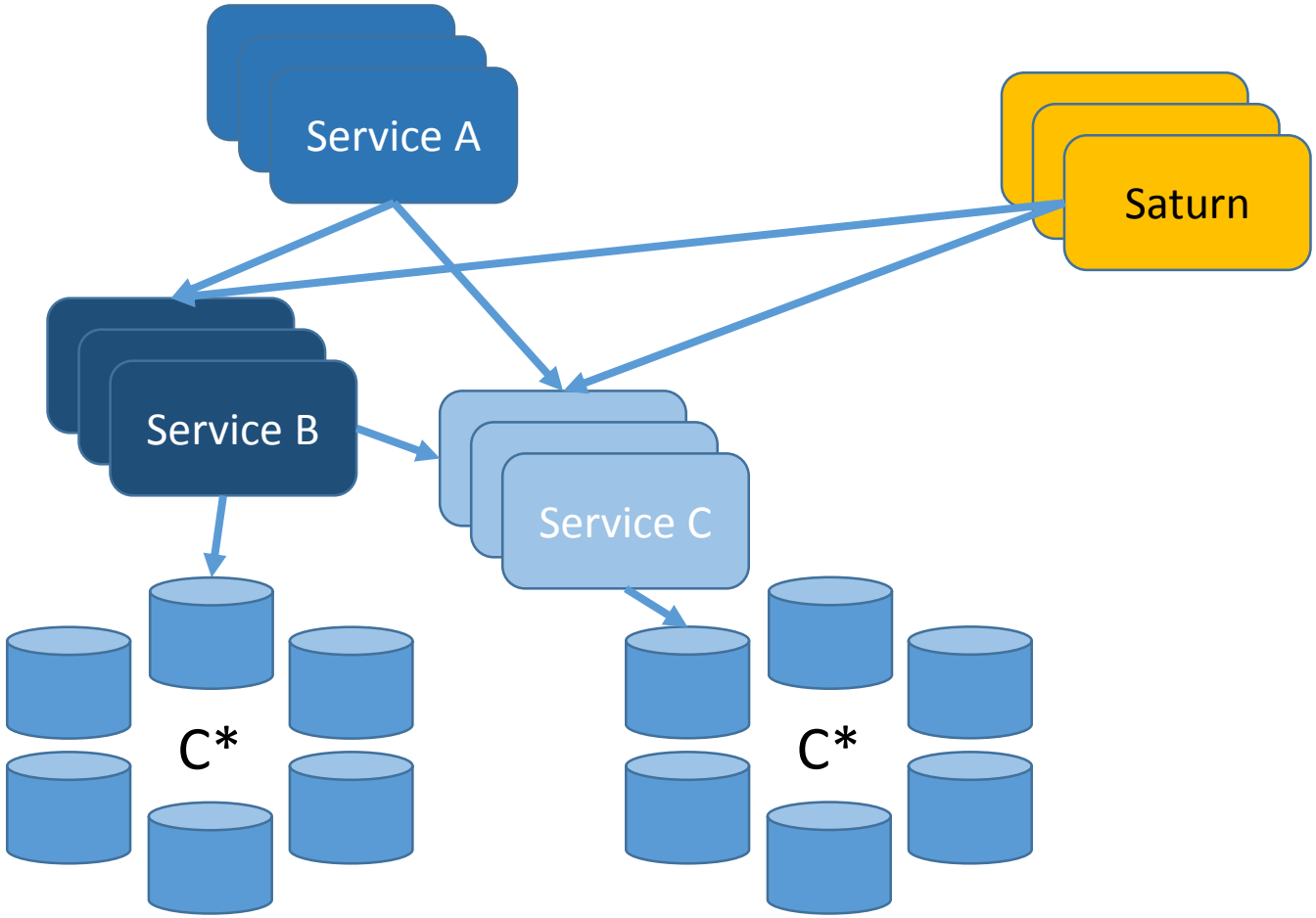
Saturn Architecture

Online Serving
Scala/micro-service architecture

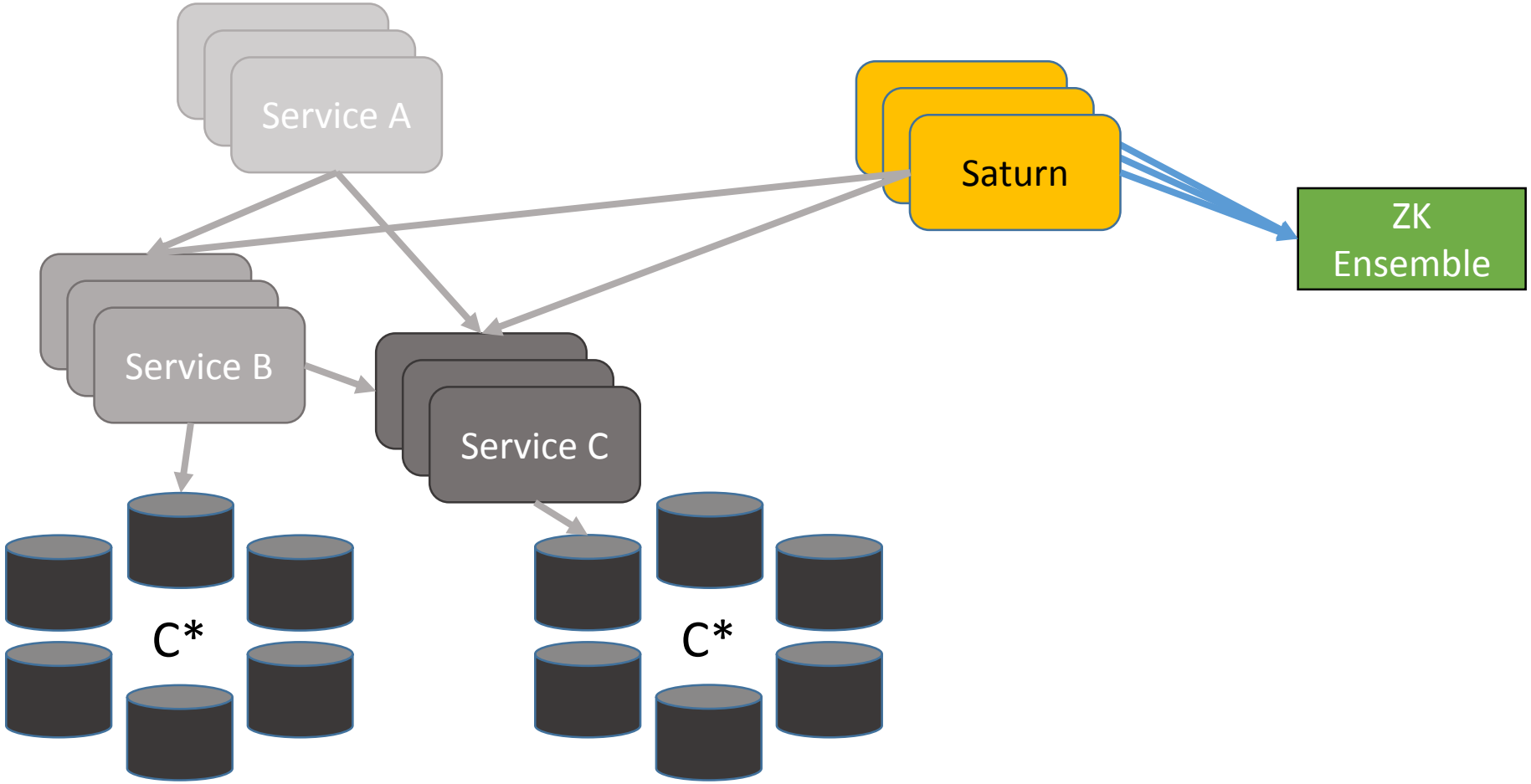


Saturn Architecture

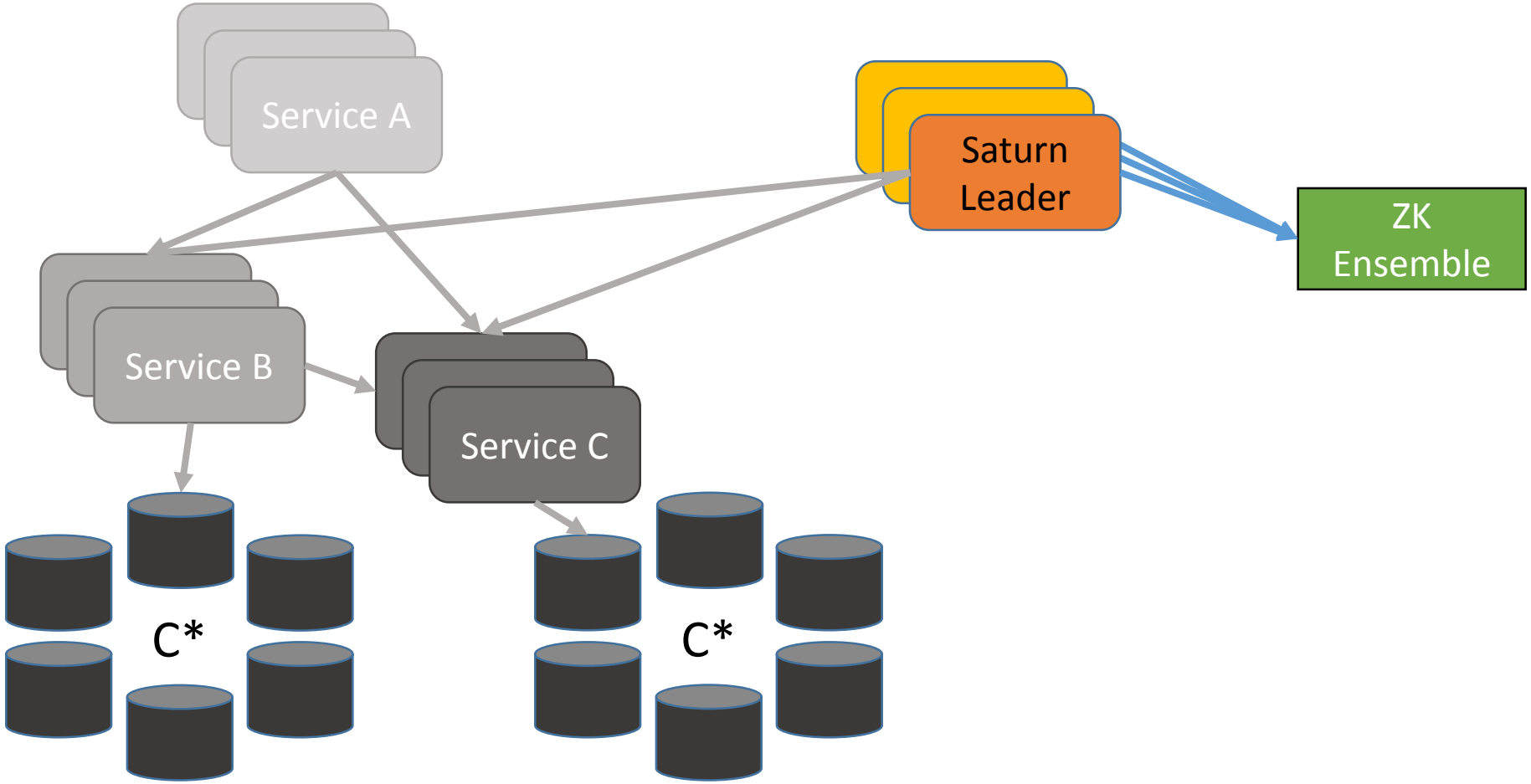
Online Serving
Scala/micro-service architecture



Saturn Architecture



Saturn Architecture

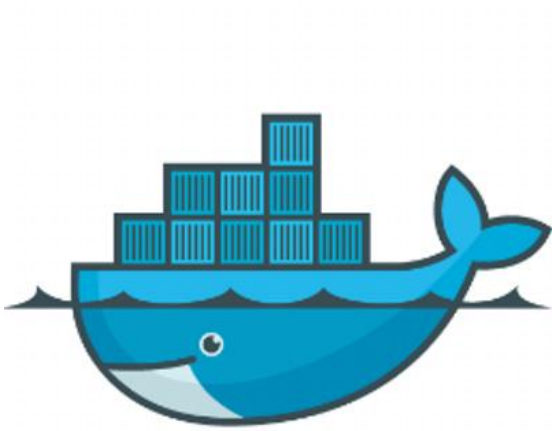


Problems with Saturn

- Single master meant naïve implementation ran all jobs in same JVM
  - Huge CPU contention @ top of the hour
  - OOM Exceptions & GC issues

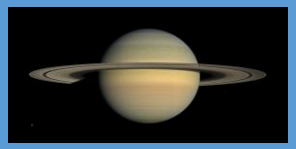
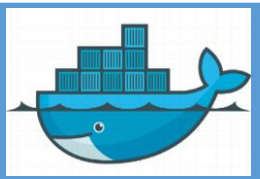
Enter: Docker

Containers allow for resource isolation!

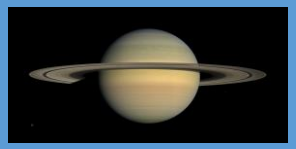
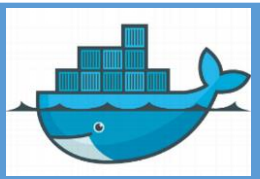


CC-by-2.0 https://www.flickr.com/photos/photohome_uk/1494590209

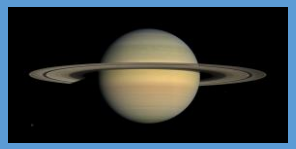
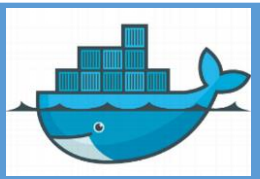
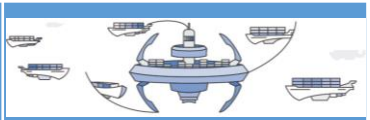
Supported Features

Platform	 Saturn	 Docker
Run code	✓	✓
Resource Isolation	✗	✓

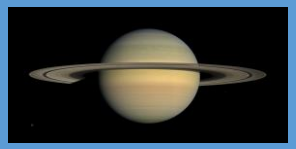
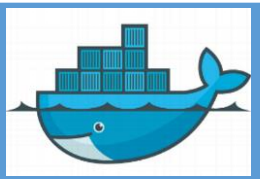
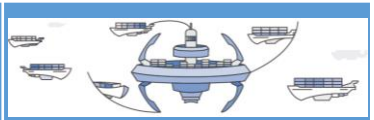
Supported Features

Platform	 Saturn	 Docker
Run code	✓	✓
Resource Isolation	✗	✓
Clusters / HA	✓	✗

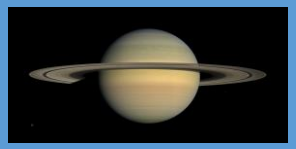
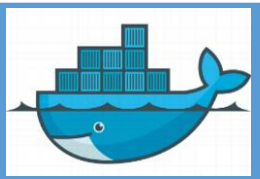
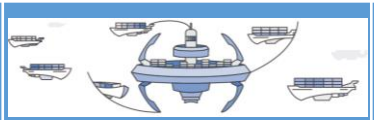
Supported Features

Platform	 Saturn	 Docker	 Amazon ECS
Run code	✓	✓	✓
Resource Isolation	✗	✓	✓
Clusters / HA	✓	✗	✓

Supported Features

Platform	 Saturn	 Docker	 Amazon ECS
Run code	✓	✓	✓
Resource Isolation	✗	✓	✓
Clusters / HA	✓	✗	✓
Great developer workflow	✓	✗	✗
Scheduled Jobs	✓	✗	✗

Supported Features

Platform	 Saturn	 Docker	 Amazon ECS	???
Run code	✓	✓	✓	✓
Resource Isolation	✗	✓	✓	✓
Clusters / HA	✓	✗	✓	✓
Great developer workflow	✓	✗	✗	✓
Scheduled Jobs	✓	✗	✗	✓

Solution: Iguazú

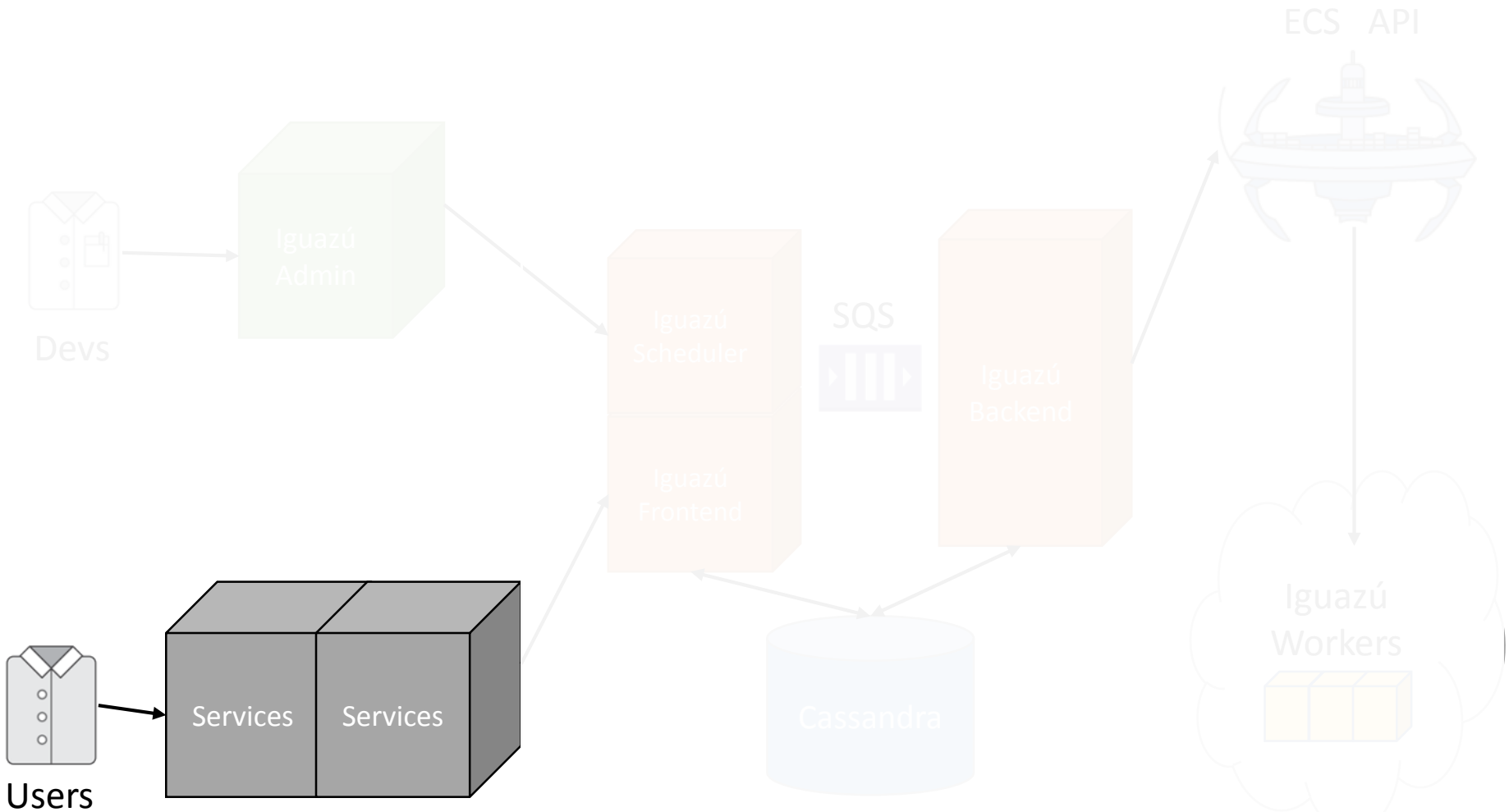
Marissa Strniste (https://www.flickr.com/photos/mstrniste/5999464924) CC-BY-2.0

Solution: Iguazú

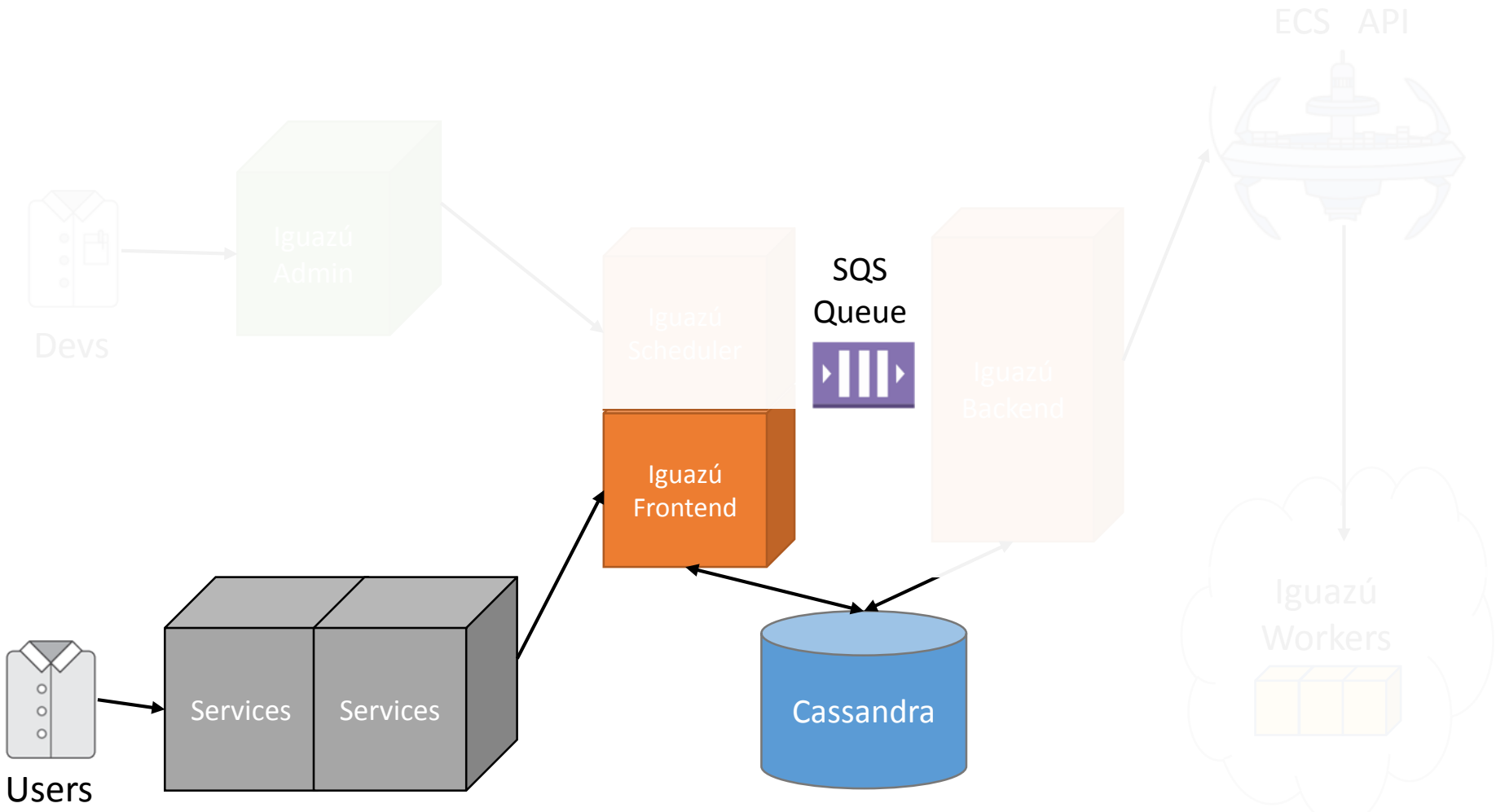
- Framework & service for asynchronous execution
  - Optimized Scala developer experience for Coursera
- Unified scheduler supports:
  - Immediate execution (nearline)
  - Scheduled recurring execution (cron-like)
  - Deferred execution (run once @ time X)

Marissa Strniste (https://www.flickr.com/photos/mstrniste/5999464924) CC-BY-2.0

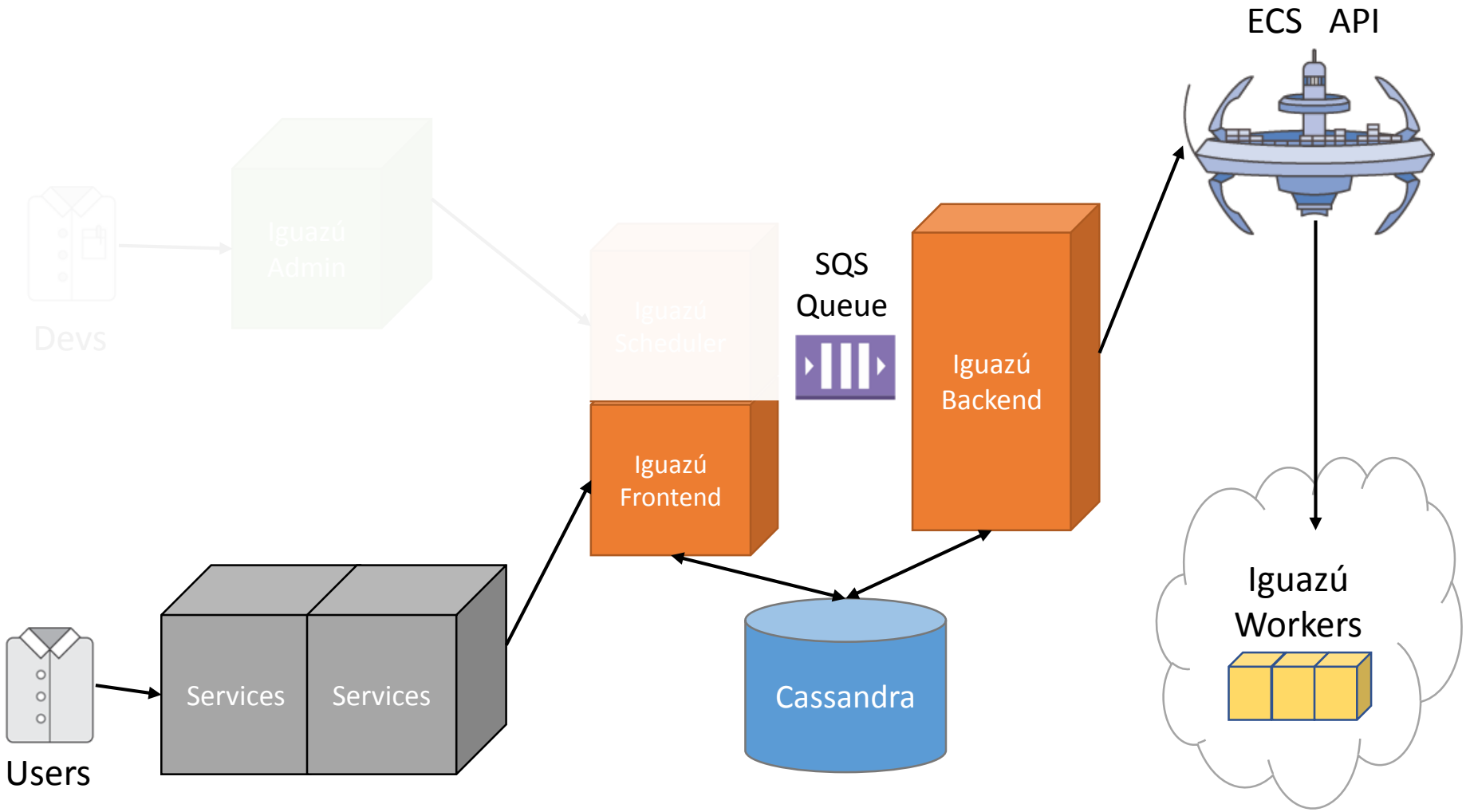
Iguazú Architecture



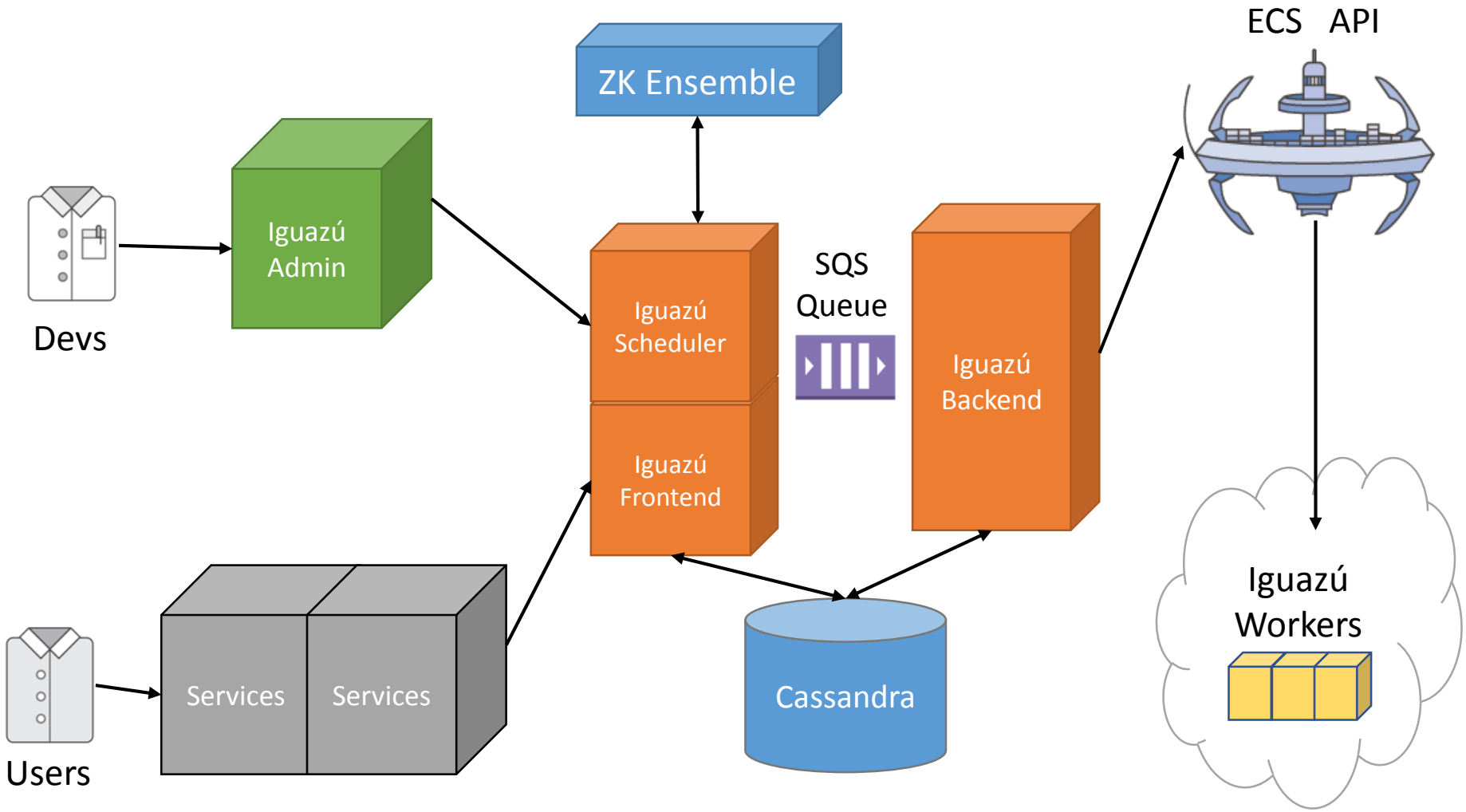
Iguazú Architecture



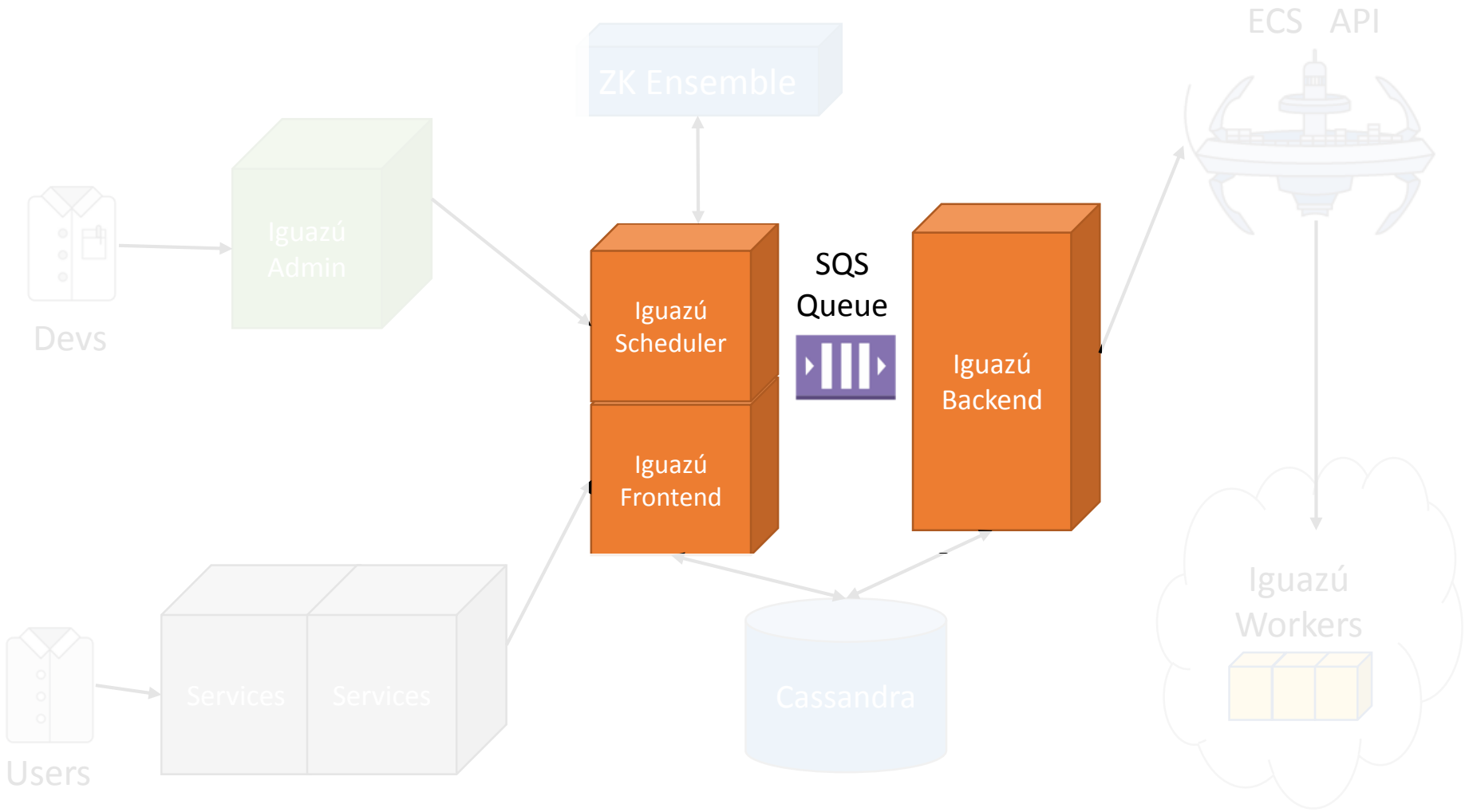
Iguazú Architecture

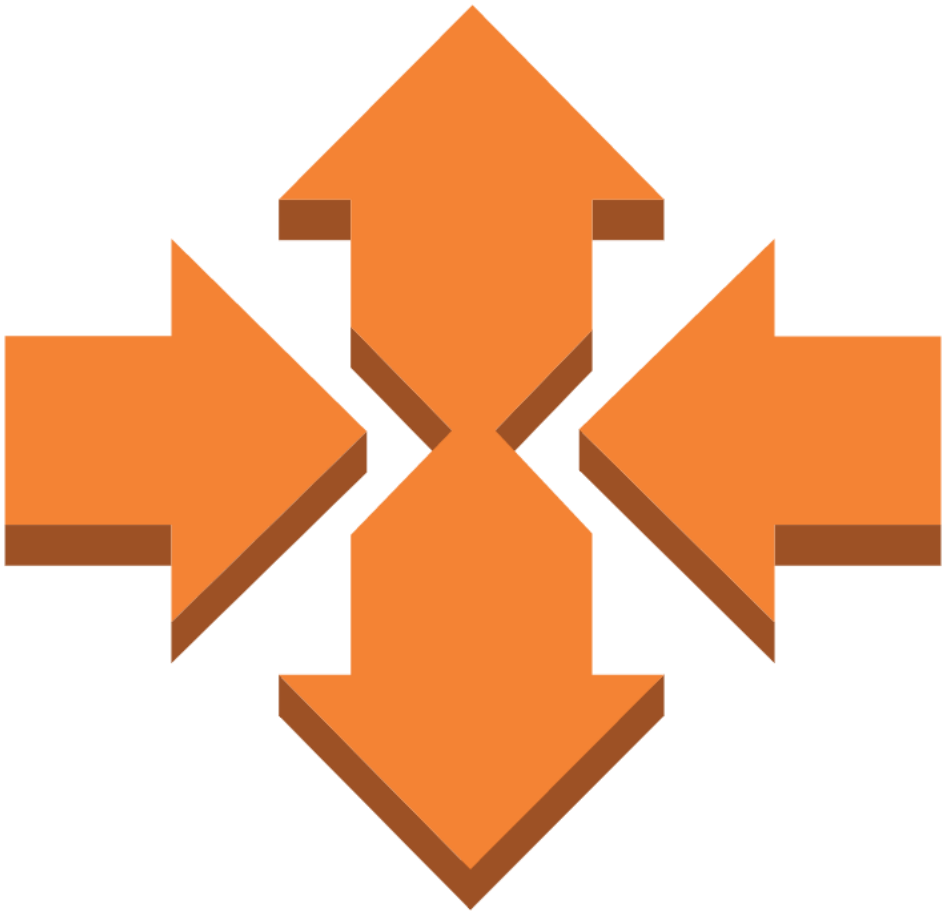


Iguazú Architecture



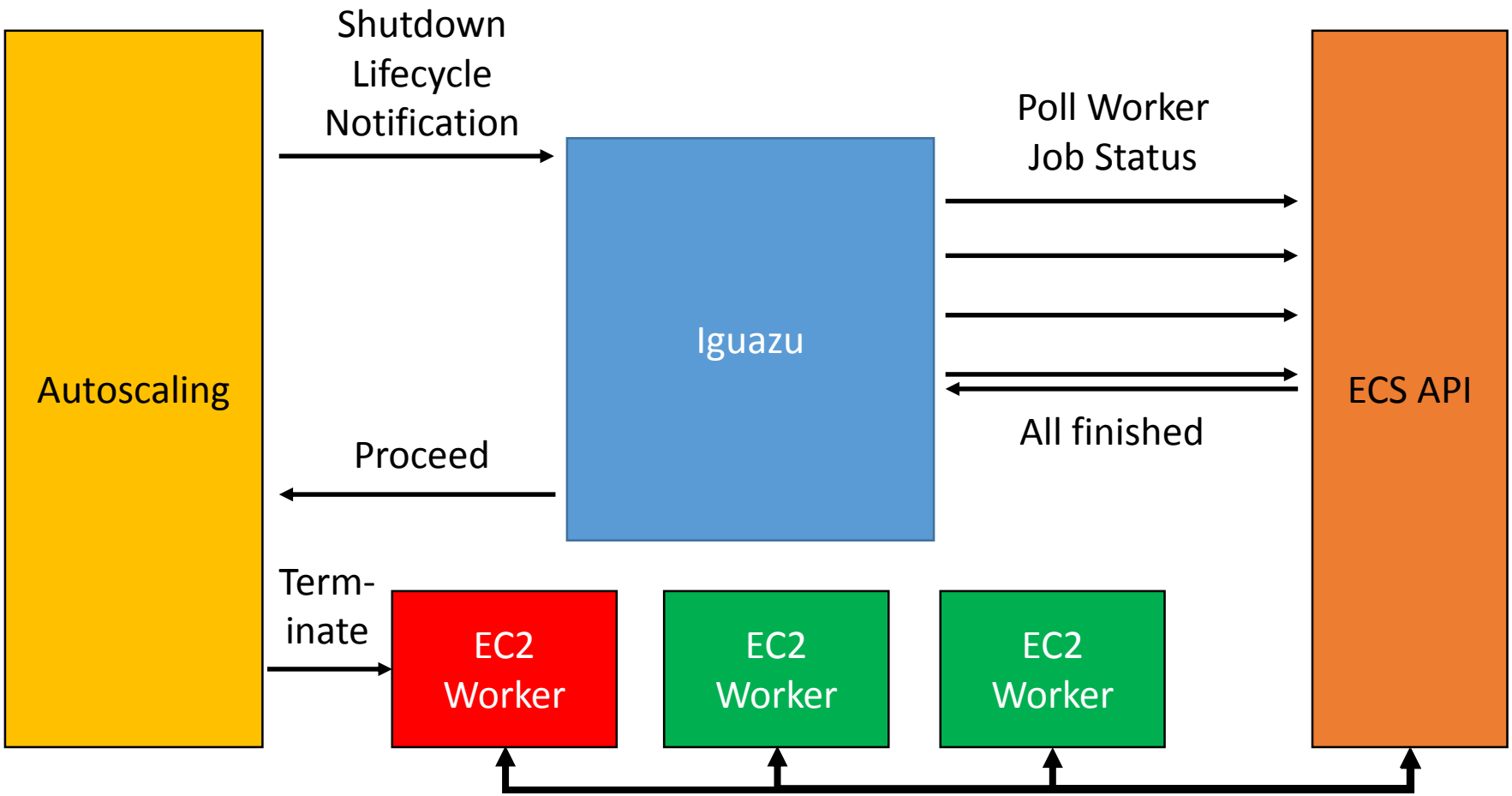
Iguazú Architecture





Autoscale, autoscale, autoscale!

Autoscaling ⇄ Iguazú ⇆ ECS



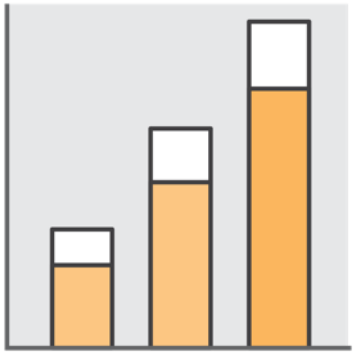
Failure in Nearline Systems

- Most jobs are non-idempotent
- Iguazú: At most once execution
  - Time-bounded delay
- Future: At least once execution
  - With caveats

Iguazú adoption by the numbers

~100 jobs in production

>100 different job schedules



>1000 runs per day

Iguazú Applications

Nearline Jobs

- Pedagogical Instructor Data Exports
- System Integrations
  - Course Migrations

Scheduled Recurring Jobs

- Course Reminders
- System Integrations
  - Payment reconciliation
  - Course translations
- Housekeeping
  - Build artifact archival
  - A/B Experiments

While containers may help you on your journey, they are not themselves a destination.

CC-by-2.0 https://www.flickr.com/photos/usoceangov/5369581593

Writing an Iguazu Job

```
class AbReminderJob @Inject() (abClient: AbClient, email: EmailAPI)
    extends AbstractJob {
  override val reservedCpu = 1024 // 1 CPU core
  override val reservedMemory = 1024 // 1 GB RAM

  def run(parameters: JsValue) = {
    val experiments = abClient.findForgotten()
    logger.info(s"Found ${experiments.size} forgotten experiments.")
    experiments.foreach { experiment =>
      sendReminder(experiment.owners, experiment.description)
    }
  }
}
```

Writing an Iguazu Job

```
class AbReminderJob @Inject() (abClient: AbClient, email: EmailAPI)
    extends AbstractJob {
  override val reservedCpu = 1024 // 1 CPU core
  override val reservedMemory = 1024 // 1 GB RAM

  def run(parameters: JsValue) = {
    val experiments = abClient.findForgotten()
    logger.info(s"Found ${experiments.size} forgotten experiments.")
    experiments.foreach { experiment =>
      sendReminder(experiment.owners, experiment.description)
    }
  }
}
```

Writing an Iguazu Job

```
class AbReminderJob @Inject() (abClient: AbClient, email: EmailAPI)
    extends AbstractJob {
  override val reservedCpu = 1024 // 1 CPU core
  override val reservedMemory = 1024 // 1 GB RAM

  def run(parameters: JsValue) = {
    val experiments = abClient.findForgotten()
    logger.info(s"Found ${experiments.size} forgotten experiments.")
    experiments.foreach { experiment =>
      sendReminder(experiment.owners, experiment.description)
    }
  }
}
```

Writing an Iguazu Job

```
class AbReminderJob @Inject() (abClient: AbClient, email: EmailAPI)
    extends AbstractJob {
  override val reservedCpu = 1024 // 1 CPU core
  override val reservedMemory = 1024 // 1 GB RAM

  def run(parameters: JsValue) = {
    val experiments = abClient.findForgotten()
    logger.info(s"Found ${experiments.size} forgotten experiments.")
    experiments.foreach { experiment =>
      sendReminder(experiment.owners, experiment.description)
    }
  }
}
```

Writing an Iguazu Job

```
class AbReminderJob @Inject() (abClient: AbClient, email: EmailAPI)
    extends AbstractJob {
  override val reservedCpu = 1024 // 1 CPU core
  override val reservedMemory = 1024 // 1 GB RAM

  def run(parameters: JsValue) = {
    val experiments = abClient.findForgotten()
    logger.info(s"Found ${experiments.size} forgotten experiments.")
    experiments.foreach { experiment =>
      sendReminder(experiment.owners, experiment.description)
    }
  }
}
```

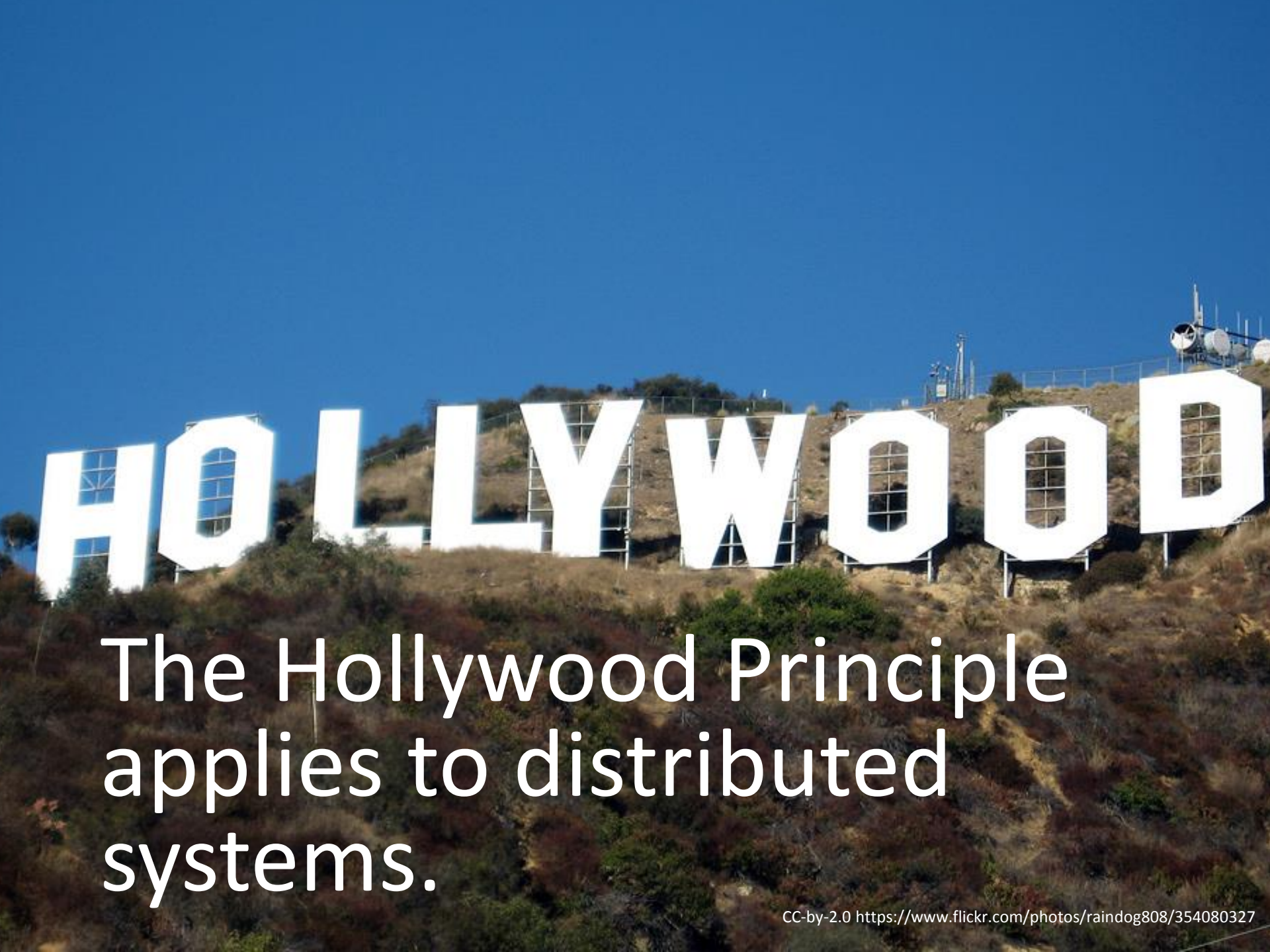
Testing an Iguazu job

infra-services — java — 88×26

```
>
s/front-page/jade/app.html" failed (2: No such file or directory), client: 192.168.56.101, server: site.dev-coursera.org, request: "GET /web/pages/front-p

in 0.000 sec

s/front-page/jade/app.html" failed (2: No such file or directory), client: 192.168.56.101, server: site.dev-coursera.org, request: "GET /web/pages/front-p

in 0.000 sec

s/front-page/jade/app.html" failed (2: No such file or directory), client: 192.168.56.101, server: site.dev-coursera.org, request: "GET /web/pages/front-p

in 0.000 sec

vagrant@site:~$ ls /home/vagrant/base/coursera/web/static.crane/pages/
about          auth           directory-webpack   groups   internet-org      nightmare-test   peer-admin
accomplishments catalog       error               home     membership-admin  open-course      peer-cal-migration
account-settings category     financial           hut      monitor-s12n      partners         quick-questions
assess         directory      global-skills-initiative  info  my-purchases  payments         s12n
vagrant@site:~$ ls base/coursera/web/static/pages/fron
```

tack frame gets deallocated. This

Branch yifan/cv set up to track remote branch yifan/cv fr



The Hollywood Principle applies to distributed systems.

CC-by-2.0 https://www.flickr.com/photos/raindog808/354080327

Deploying a new Iguazu Job

- **Developer**
  - merge into master… done

- **Jenkins Build Steps**
  - Compile & package job JAR
  - Prepare Docker image
  - Pushes image into registry
  - Register updated job with Amazon ECS API

Invoking an Iguazú Job

```
// invoking a job with one function call
// from another service via REST framework RPC

val invocationId = iguazuJobInvocationClient
    .create(IguazuJobInvocationRequest(
        jobName = "exportQuizGrades",
        parameters = quizParams))
```

A clean environment increases reliability.

CC-by-2.0 https://www.flickr.com/photos/raindog808/354080327

Evaluating Programming Assignments

An application of Iguazú

Assignment: Java Programming

You have not submitted. You must earn 80/100 points to pass.

Instructions | My submission | Discussions

There are two parts to this assignment.

Part 1 - Factorize a number: Your aim is to write a class (in Java) named 'Factoring' which takes a list of numbers via stdin and outputs all the factors of that number to stdout. Factors for each number should be returned in a single line where each factor is separated by a whitespace in increasing order.

Example Test case:

```
Input:
18
4
21

Output:
1 2 3 6 9 18
1 2 4
1 3 7 21
```

For evaluation, your code will be run against our test cases to see if they pass. You'll receive full grade only if all the test cases pass.

Part 2 : Find if a number is prime - Given a list of numbers, your program should output whether each number is a prime or not. Input is received via stdin and you should output "true" or "false" based on whether the number is prime or not.

Example Test case:

How to submit

When you're ready to submit, you can upload files for each part of the assignment on the "My submission" tab.

Assignment: Java Programming

You have not submitted. You must earn 80/100 points to pass.

Instructions | My submission | Discussions

✖ Cancel

Upload Files and Submit

To upload a file, click the part below. Then, submit the files. You can submit as many times as you like. You do not need to upload all parts in order to submit.

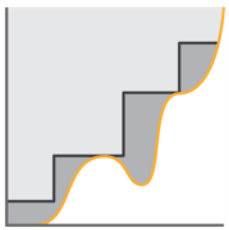
Factorize a number	
50 points	Factoring.java
Find if a number is prime	
50 points	Prime.java

Submit

Your Submissions

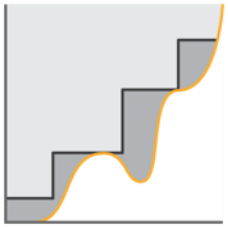
Graded submissions will appear here

Design Goals



Elastic
Infrastructure

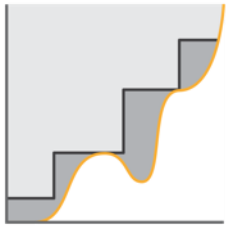
Design Goals



Elastic
Infrastructure

No
Maintenance

Design Goals



Elastic Infrastructure

No Maintenance

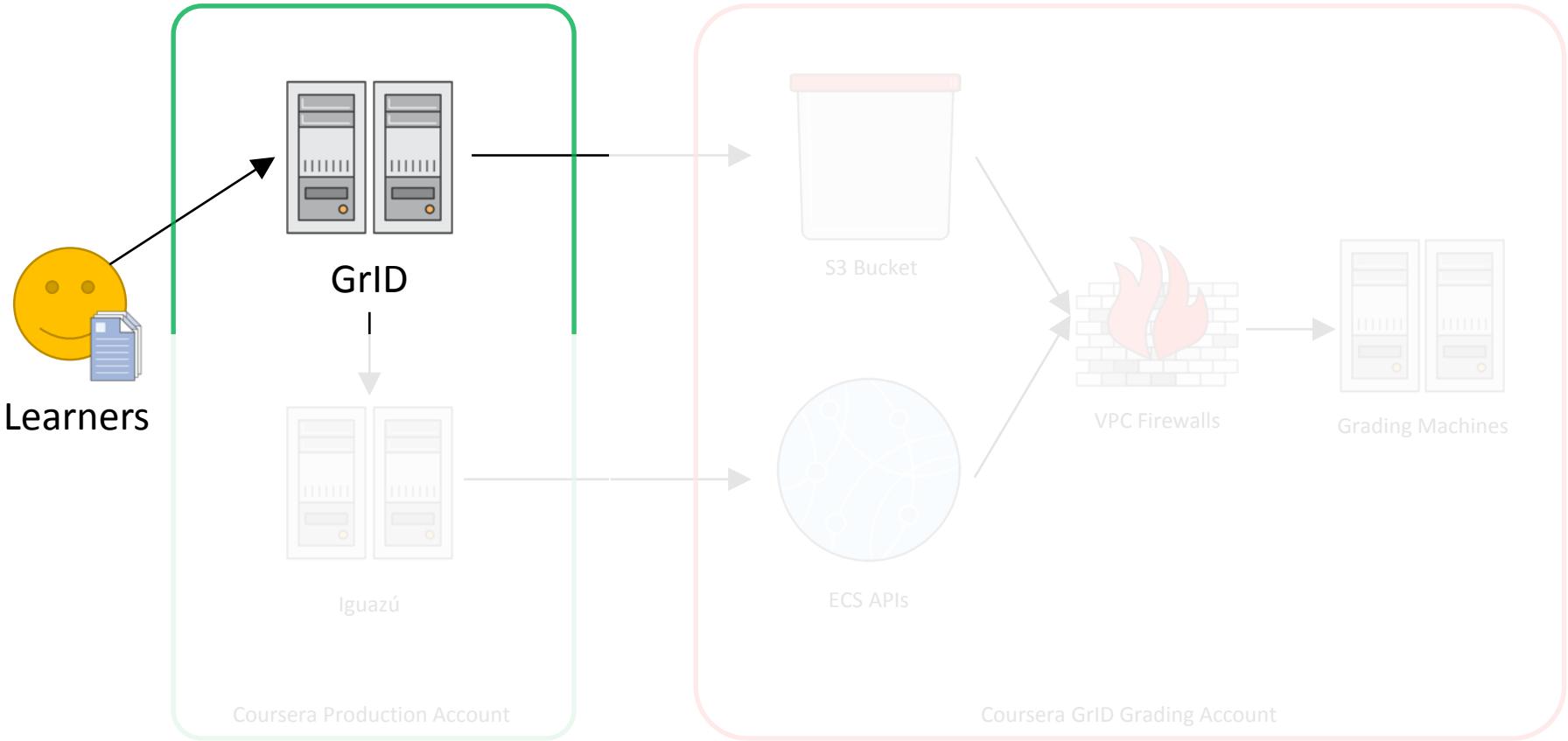
Near Real-time

Solution: **GrID**

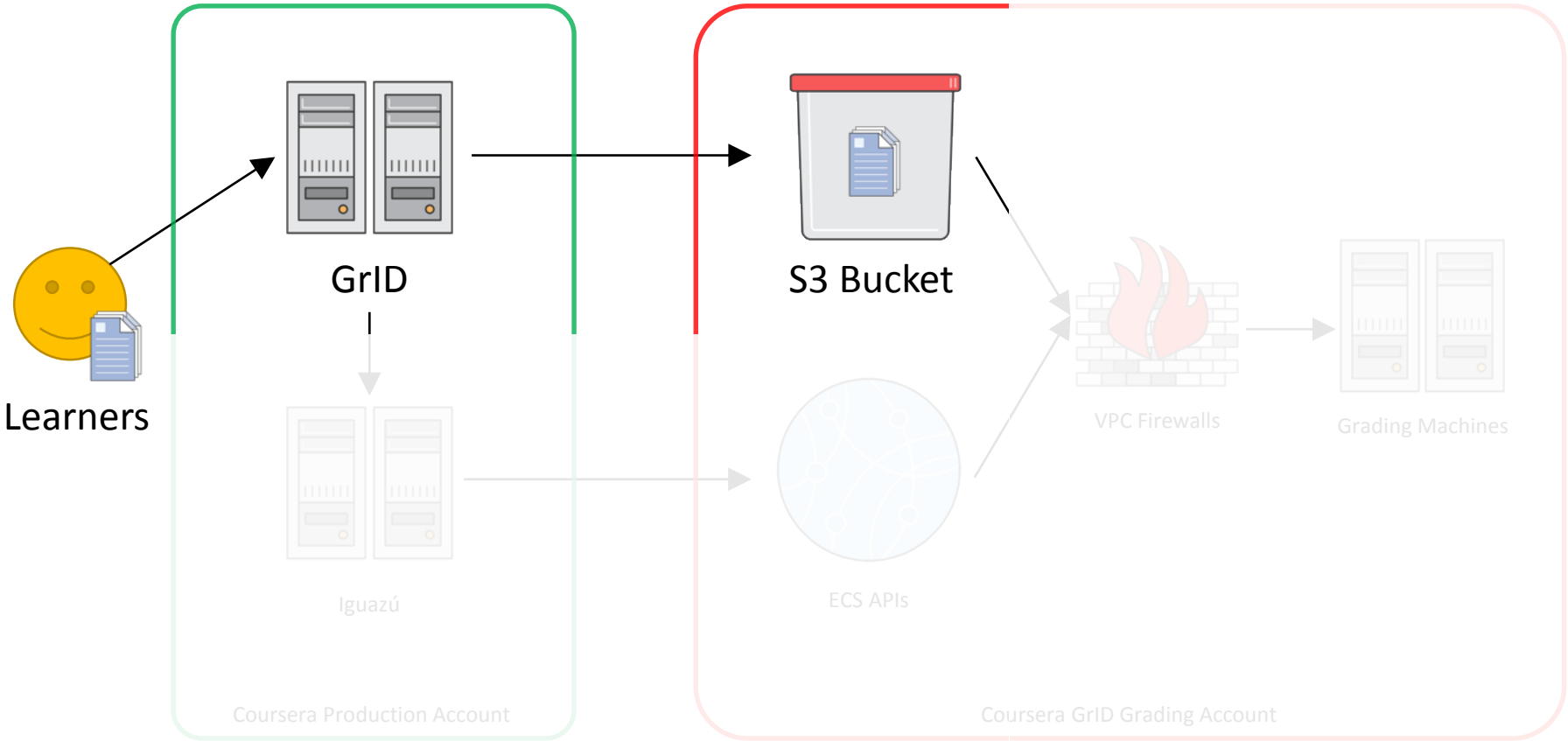
- Service + framework for grading programming assignments
- Builds on Iguazú
- Named for Tron's "digital frontier"
  - Backronym: **Gr**ading **I**nside **D**ocker

Patrick Hoesly (https://www.flickr.com/photos/zooboing/5665221326/) CC-BY-2.0

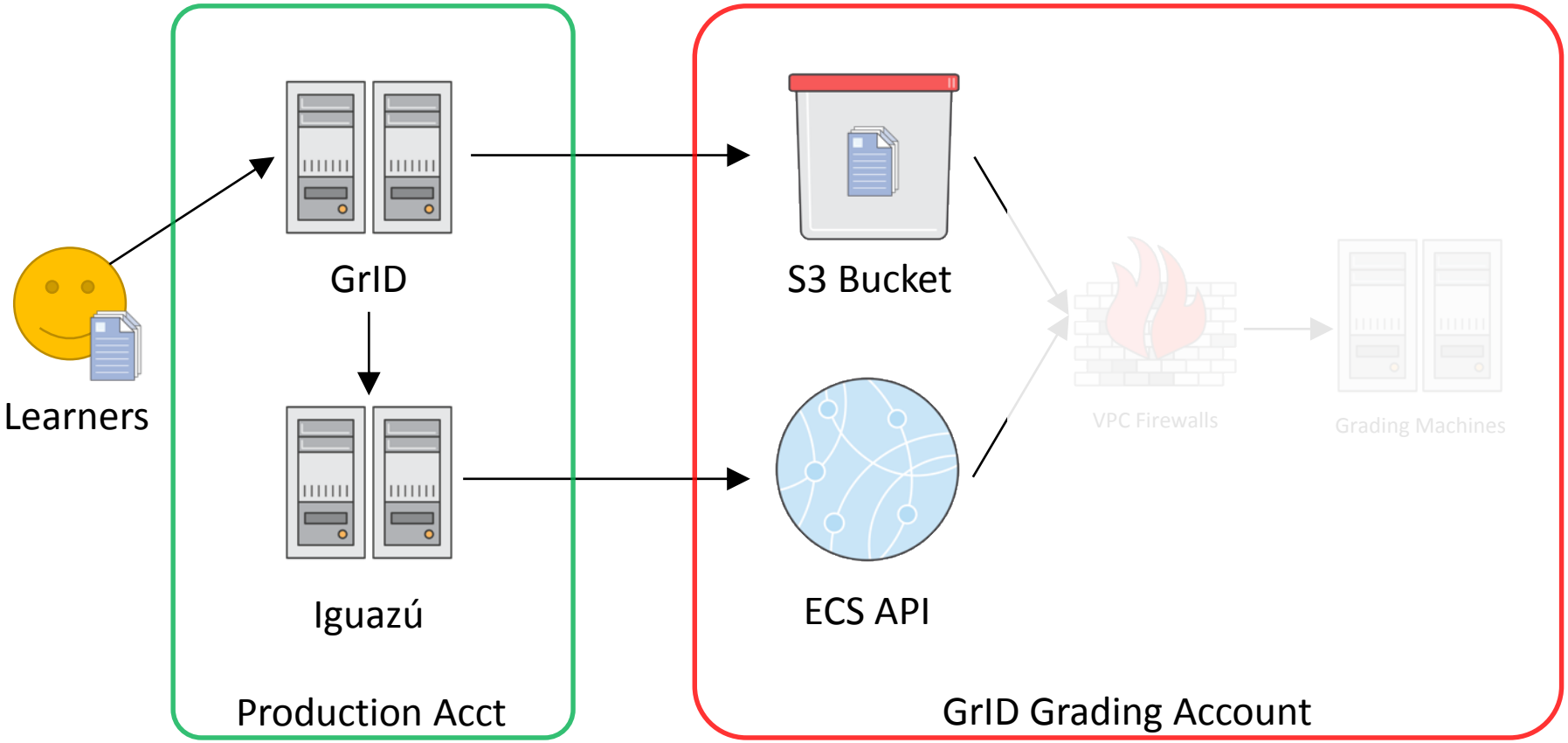
High-level GrID Architecture



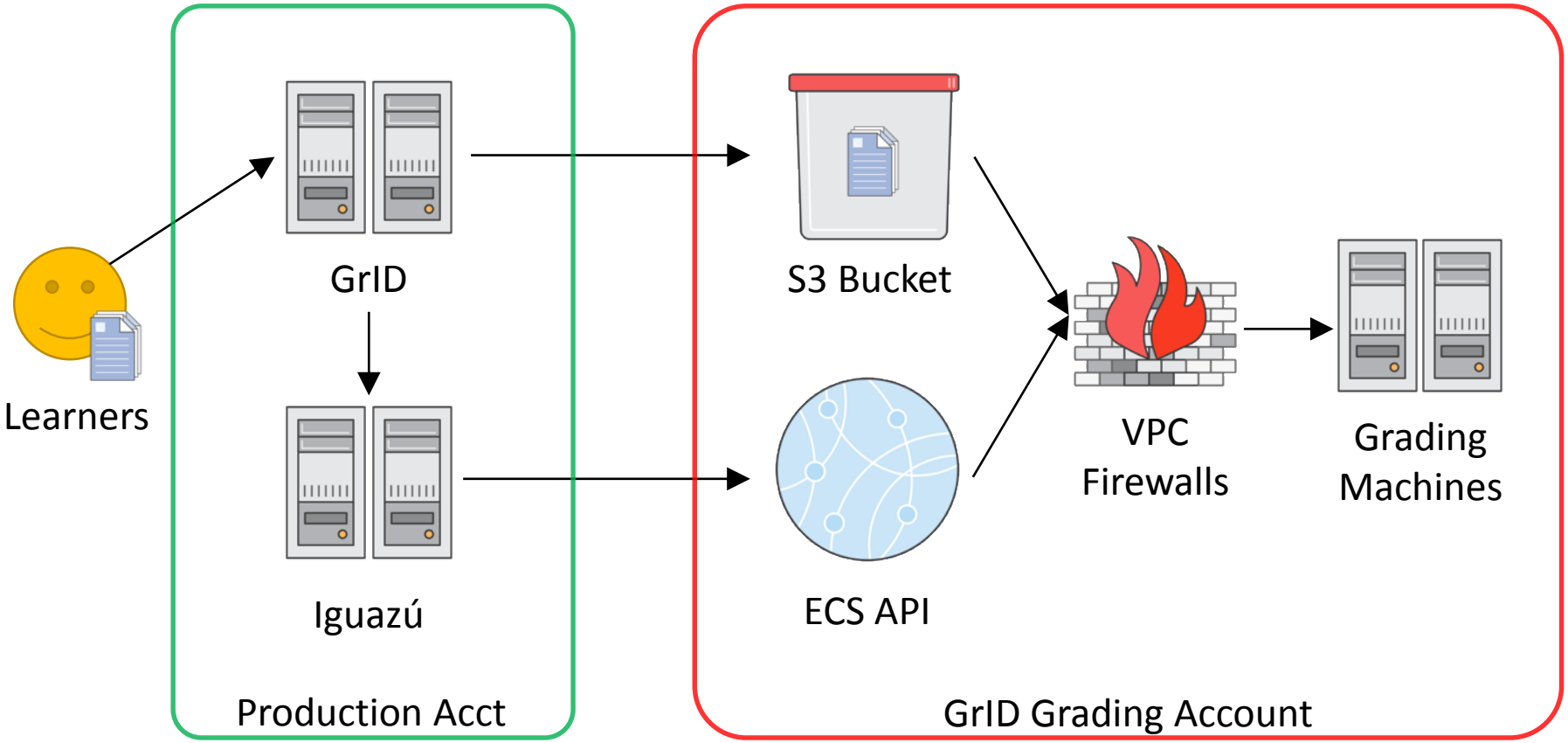
High-level GrID Architecture



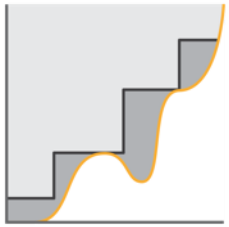
High-level GrID Architecture



High-level GrID Architecture



Design Goals



Elastic Infrastructure

No Maintenance

Near Real-time

Secure Infrastructure

Programming Assignments

The Security Challenge

Compiling and running **untrusted, arbitrary code** on our cluster in near real time.

Would you like to compile and run C code from random people on the Internet on your servers?

```
FROM redis

FROM ubuntu:latest

FROM jane’s-image
```

Security Assumptions

- Run arbitrary binaries
- Instructor grading scripts may have vulnerabilities
  - ∴ Grading code is untrusted
- Unknown vulnerabilities in Docker and Linux name-spacing and/or container implementation

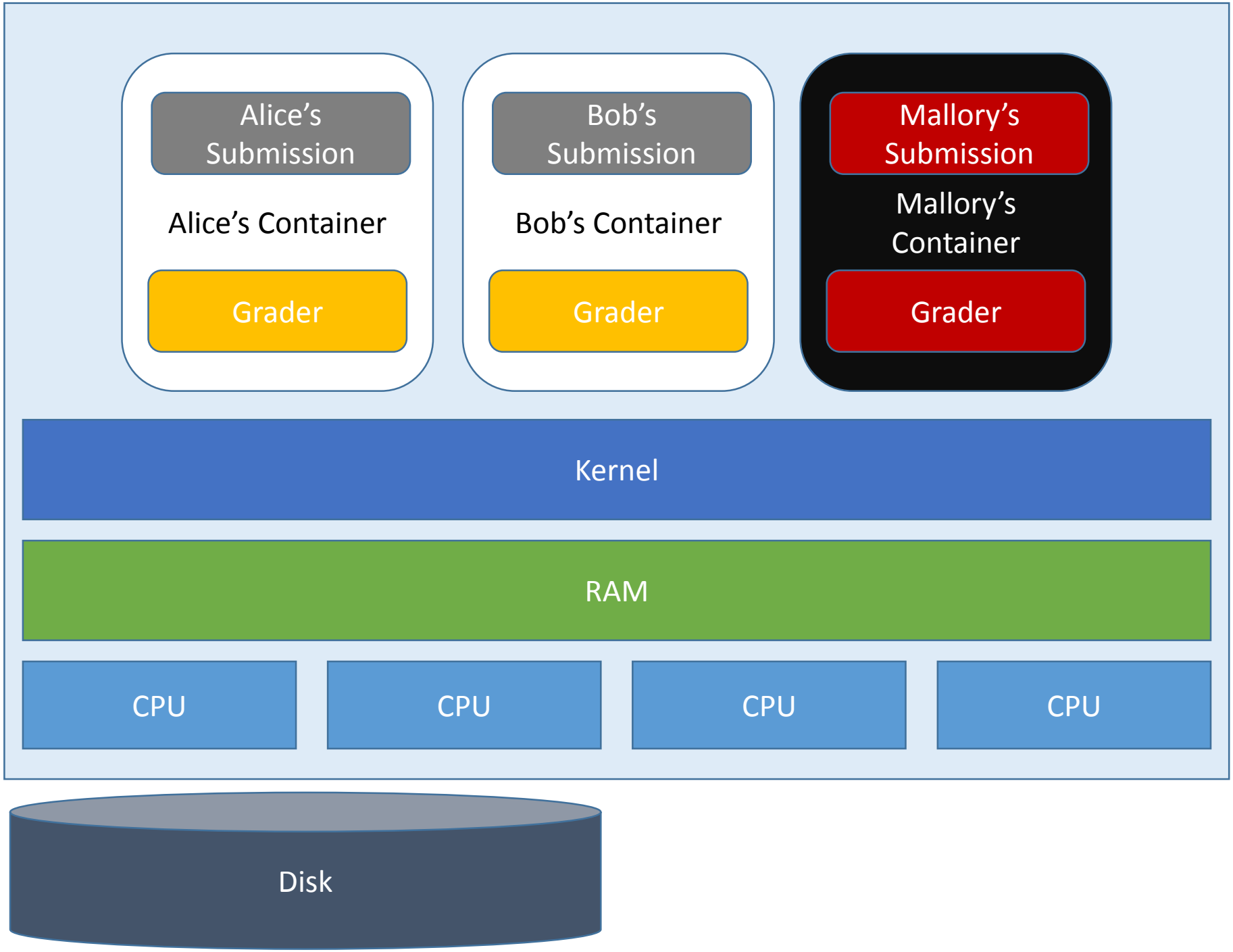
Security Goals

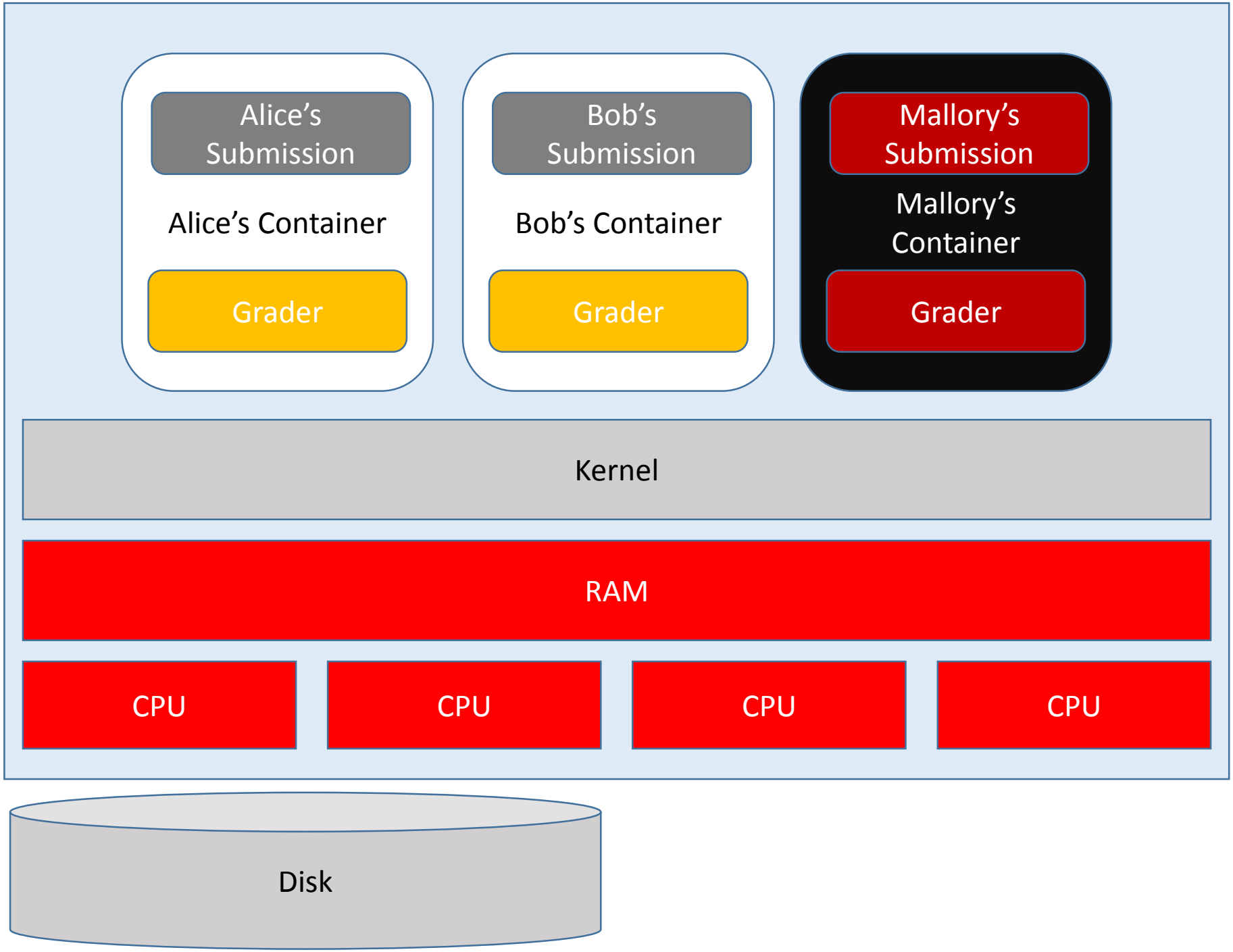
Prevent submitted code from:

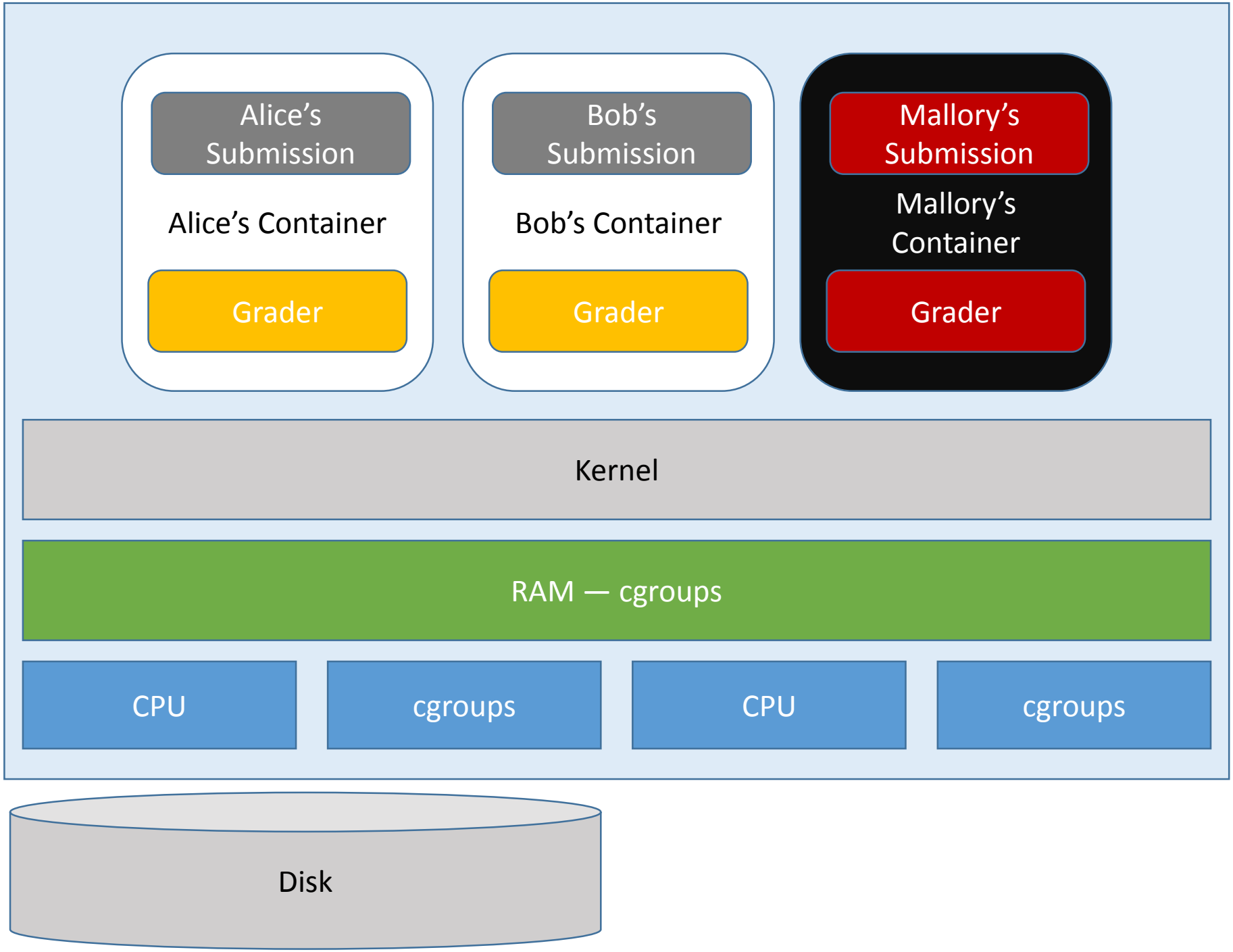
- impacting the evaluation of *other* submissions.
- disrupting the grading environment (e.g., DoS)
- affecting the rest of the Coursera learning platform

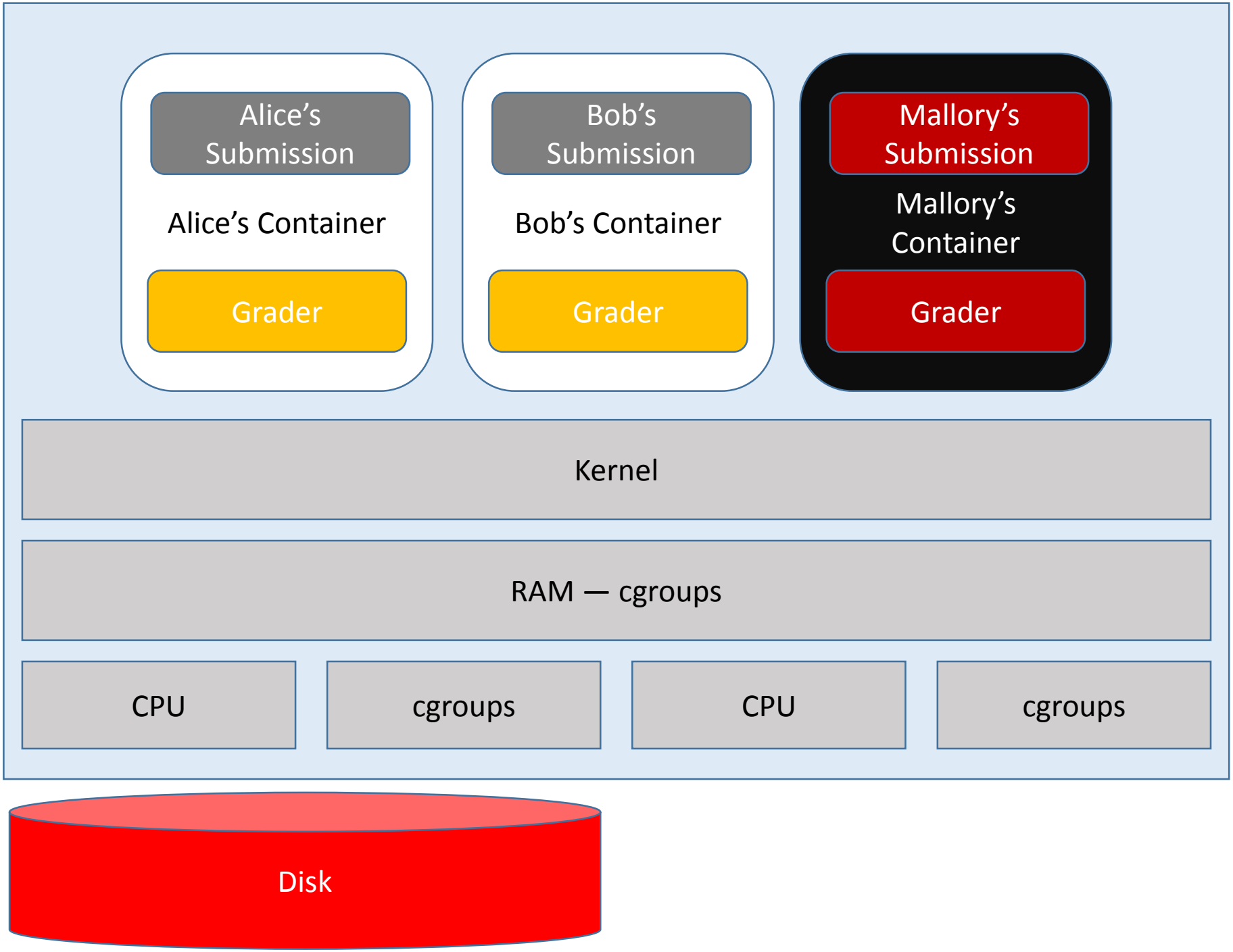
Grading assignment submissions

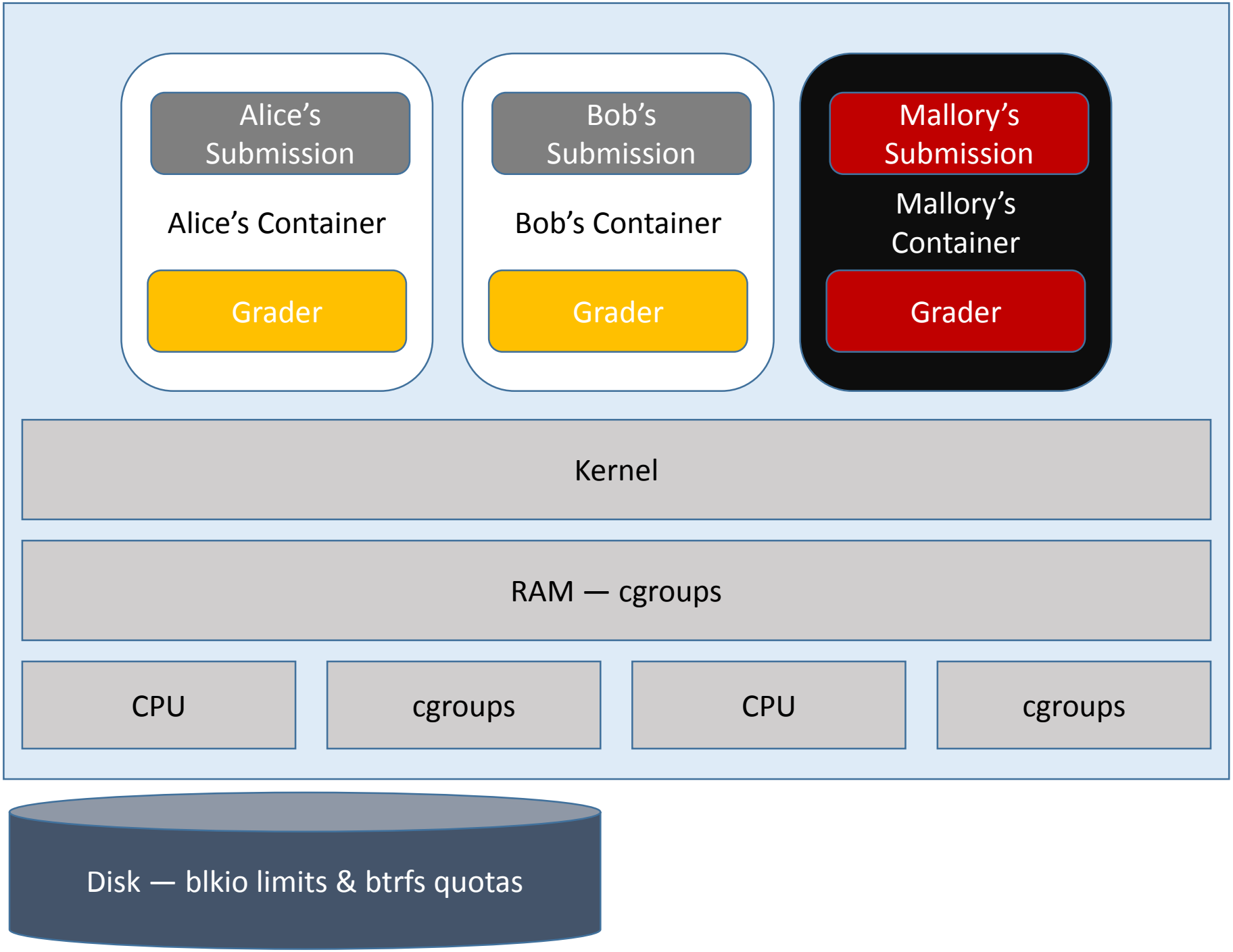
CC-by-2.0 https://www.flickr.com/photos/dherholz/4367511580/

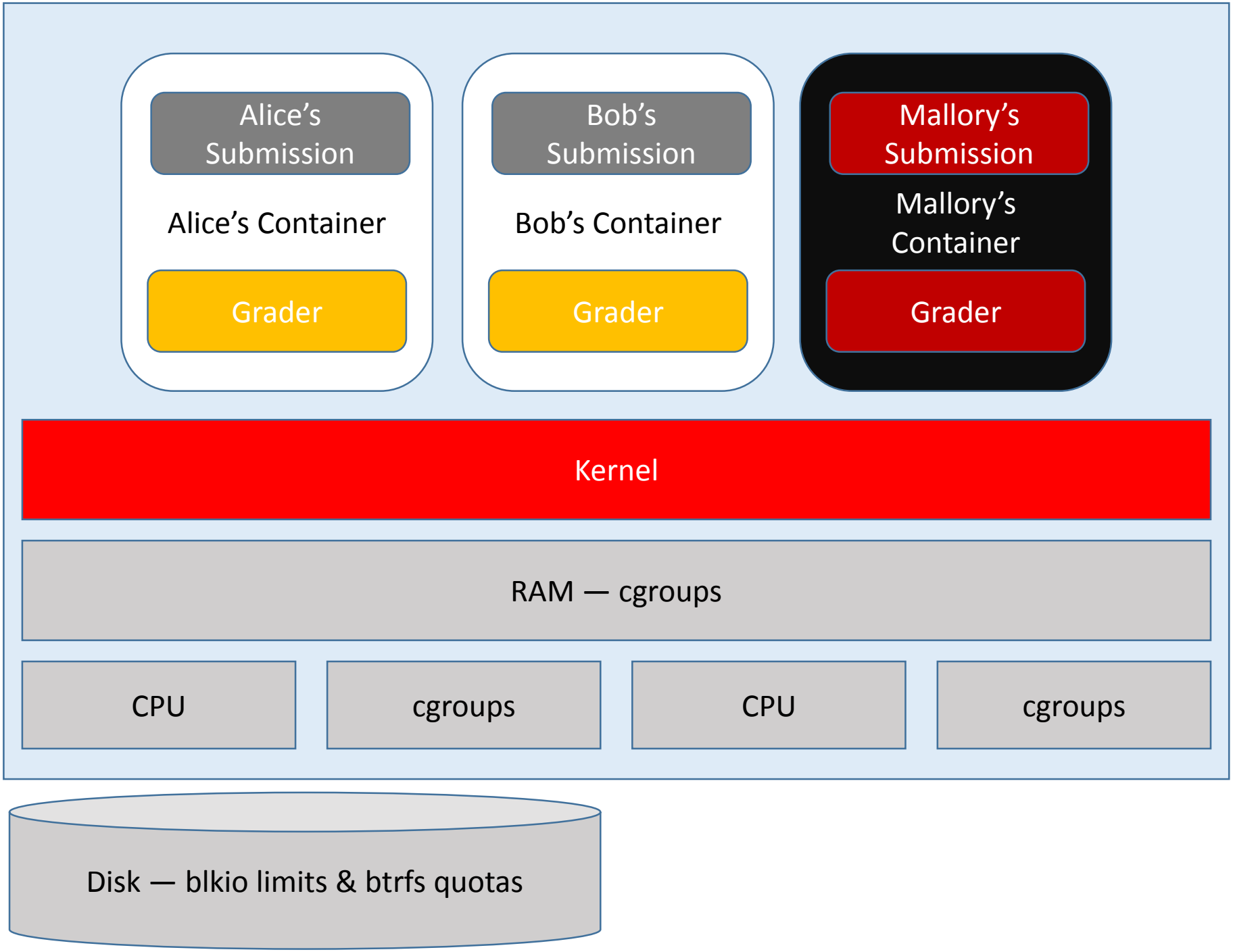

Alice's Submission
Alice's Container
Grader
Bob's Submission
Bob's Container
Grader
Mallory's Submission
Mallory's Container
Grader
Kernel
RAM
CPU
CPU
CPU
CPU
Disk


Alice's Submission
Alice's Container
Grader
Bob's Submission
Bob's Container
Grader
Mallory's Submission
Mallory's Container
Grader
Kernel
RAM
CPU
CPU
CPU
CPU
Disk

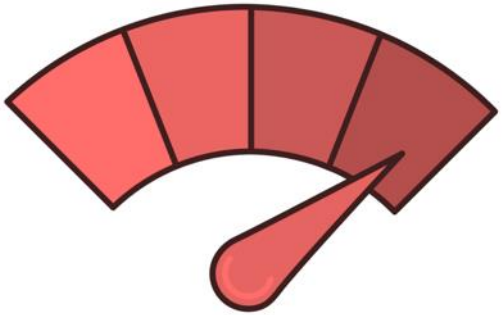

Alice's Submission
Alice's Container
Grader
Bob's Submission
Bob's Container
Grader
Mallory's Submission
Mallory's Container
Grader
Kernel
RAM — cgroups
CPU
cgroups
CPU
cgroups
Disk


Alice's Submission
Alice's Container
Grader
Bob's Submission
Bob's Container
Grader
Mallory's Submission
Mallory's Container
Grader
Kernel
RAM — cgroups
CPU
cgroups
CPU
cgroups
Disk

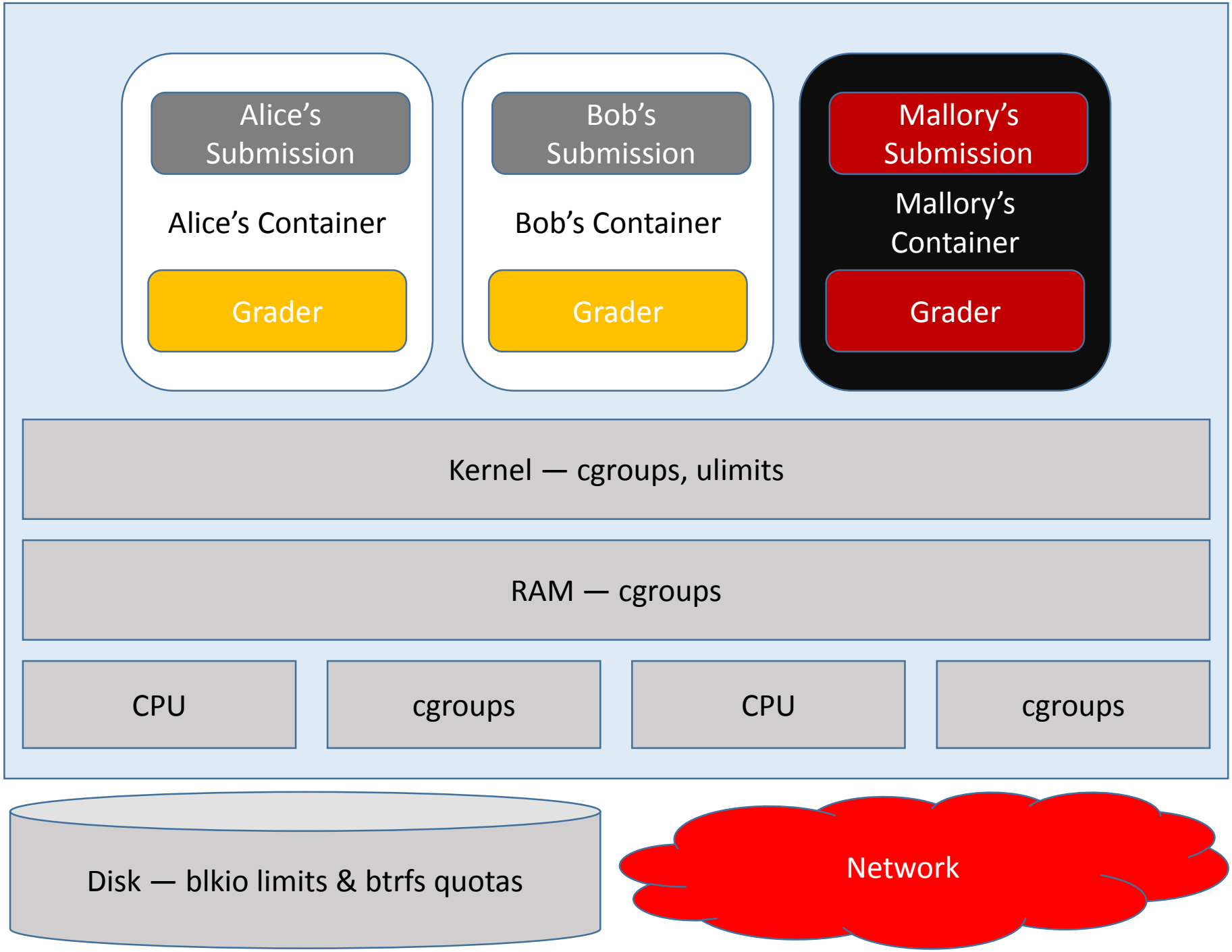

Alice's Submission
Alice's Container
Grader
Bob's Submission
Bob's Container
Grader
Mallory's Submission
Mallory's Container
Grader
Kernel
RAM — cgroups
CPU
cgroups
CPU
cgroups
Disk — blkio limits & btrfs quotas


Alice's Submission
Alice's Container
Grader
Bob's Submission
Bob's Container
Grader
Mallory's Submission
Mallory's Container
Grader
Kernel
RAM — cgroups
CPU
cgroups
CPU
cgroups
Disk — blkio limits & btrfs quotas

Attacks: Kernel Resource Exhaustion



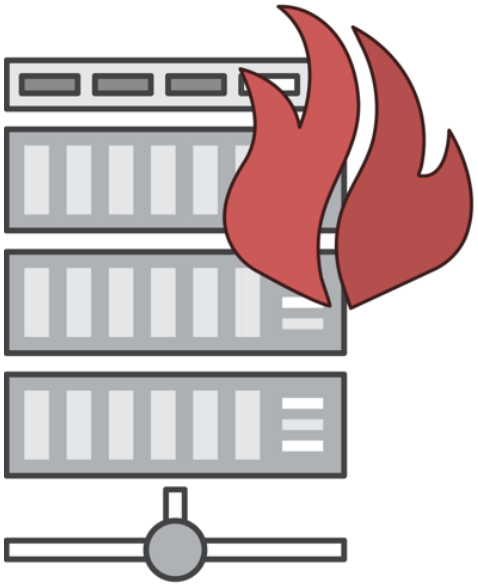
- Open file limits per container (`nofile`)
- `nproc` Process limits
- Limit kernel memory per `cgroup`
- Limit execution time


Alice's Submission
Alice's Container
Grader
Bob's Submission
Bob's Container
Grader
Mallory's Submission
Mallory's Container
Grader
Kernel — cgroups, ulimits
RAM — cgroups
CPU
cgroups
CPU
cgroups
Disk — blkio limits & btrfs quotas
Network

Attacks: Network attacks

Attacks:

- Bitcoin mining
- DoS attacks on other systems
- Access Amazon S3 and other AWS APIs



Defense:

- Deny network access

Docker Network Modes

`NetworkDisabled` too restrictive

- Some graders require local loopback
- Feature also deprecated

`--net=none + deny net_admin + audit network`

- Isolation via Docker creating an independent network stack for each container

github.com/coursera/amazon-ecs-agent

CC-by-2.0 https://www.flickr.com/photos/valentinap/253659858

CC-by-2.0 https://www.flickr.com/photos/jessicafm/2834658255/

CC-by-2.0 https://www.flickr.com/photos/donnieray/11501178306/in/photostream/

Defense in Depth

- Mandatory Access Control (App Armor)
  - Allows auditing or denying access to a variety of subsystems
- Drop capabilities from bounding set
  - No need for `NET_BIND_SERVICE, CAP_FOWNER, MKNOD`
- Deny root within container

Deny Root Escalations

- We modify instructor grader images before allowing them to be run
  - Clears setuid
  - Inserts C wrapper to drop privileges from root and redirect stdin/stdout/stderr
- Run cleaning job on another Iguazú cluster
  - Run Docker in Docker!
- Docker 1.10 adds User Namespaces

If all else fails…

- Utilizes VPC security measures to further restrict network access
  - No public internet access
  - Security group to restrict inbound/outbound access
  - Network flow logs for auditing
- Separate AWS account
- Run in an Auto Scaling group
  - Regularly terminate all grading EC2 instances

Other Security Measures

- Utilize AWS CloudTrail for audit logs
- Third-party security monitoring (Threat Stack)
  - No one should log in, so any TTY is an alert
- Penetration testing by third-party red team (Synack)

Lessons Learned - GrID

- Building a platform for code execution is hard!
- Carefully monitor disk usage
- Run the latest kernels
  - Latest security patches
  - `btrfs` wedging on older kernels
    - Default Ubuntu 14.04 kernel not new enough!


Reliable deploy
tooling pays for itself.

coursera

Thank you!

Brennan Saeta
github/saeta
@bsaeta
saeta@coursera.org

GrID lead

QCon LONDON

Frank Chen
github/frankchn
@frankchn
frankchn@coursera.org

Iguazú Lead

coursera

Questions?

Brennan Saeta
github/saeta
@bsaeta
saeta@coursera.org

GrID lead

QCon LONDON

Frank Chen
github/frankchn
@frankchn
frankchn@coursera.org

Iguazú Lead