amazon web services

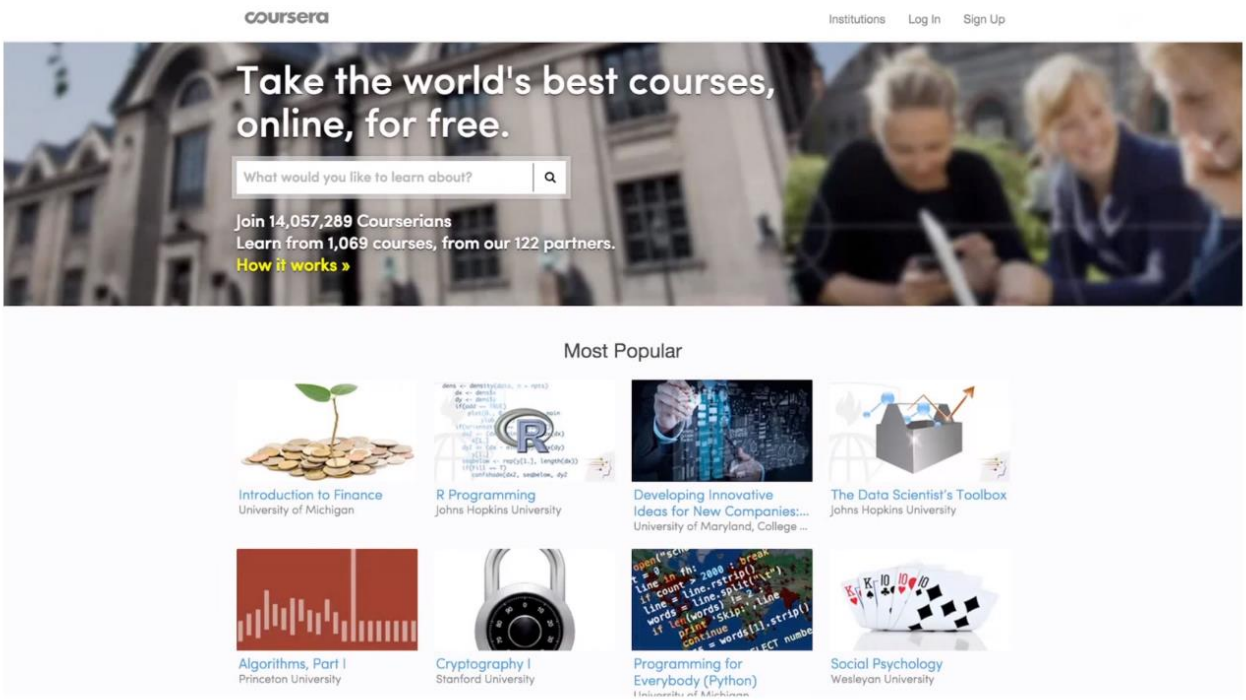
Container Day 2015

AWS Pop-up Loft - San Francisco

Container Day: Batch Processing with Amazon EC2 Container Service

Coursera: Brennan Saeta, Frank Chen

amazon web services



What We Do

Massive Open Online Courses

14 million
learners

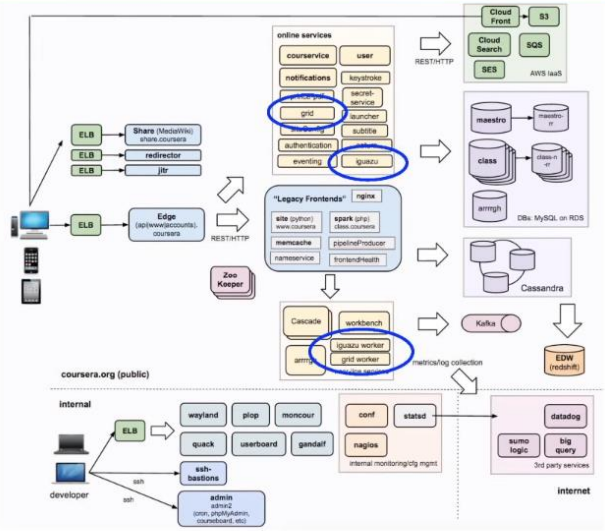
1000
courses

2.2 million
course completions

120
partners

coursera

Coursera Technical Architecture



coursera

Batch Processing Enables...

Reporting

Instructor Reports

- Grade exports
- Learner demographics
- Course progress statistics

Internal Reports

- Business metrics
- Payments reconciliation

Batch Processing Enables...

Marketing

- Recommendation emails
- Batch marketing / reactivation emails

Batch Processing Enables...

Pedagogical Innovation

- Auto-graded programming assignments
- Peer-review matching & analysis

Bad Old Days of Batch Processing

Cascade

- **PHP-based** job runner
- Run in screen sessions
- Polled for new jobs
- Fragile and unreliable
- Forced restarts on regular basis

Bad Old Days of Batch Processing II

Saturn

- **Scala-based** scheduled batch process runner
  - Powered by Quartz Scheduler library
- **Cron-only jobs**
  - Cannot run jobs on demand
  - Some jobs have to wake up frequently to poll
- All jobs ran on same instance (and JVM), causing interference

What Do We Want?

Reliability

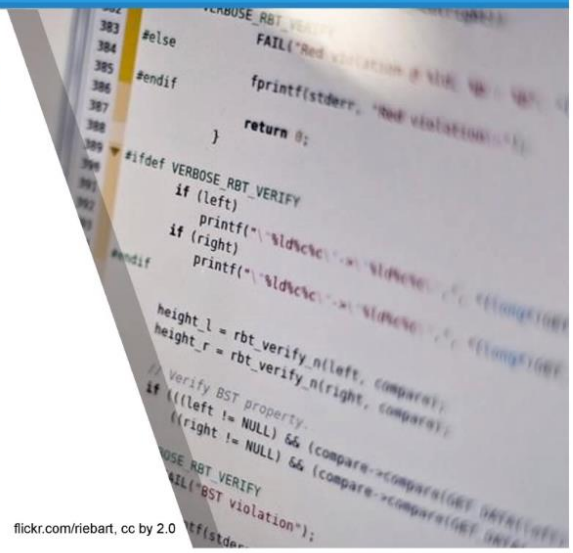
- Saturn / Cascade were flaky
- Developers became frustrated with jobs not running properly

flickr.com/rclarkeimages, cc by-nc 2.0

What Do We Want?

Easy Development

- Developing & testing locally was difficult
- Little or no boilerplate should be required



flickr.com/riebart, cc by 2.0

What Do We Want?

Easy Deployment

- Deployment was difficult and non-deterministic
- *"Other services have one-click tools, why can't your service have that too?"*

flickr.com/derelllicht, cc by-nd 2.0

What Do We Want?

High Efficiency

- Cost-conscious
- Most jobs complete < 20 minutes
  - EC2 rounds costs up to full hour
- Startup time of individual instances

flickr.com/koertmichiels, cc by-nc-nd 2.0

What Do We Want?

Low Ops Load

- Only one dev-ops engineer -- can't manage everything
- Developers own their services
- Developers shouldn't have to actively monitor services

flickr.com/reynermedia, cc by 2.0

Alternative Technologies

Home-grown Tech

- Tried, but proved to be unreliable
- Difficult to handle coordination and synchronization

MESOS

- Very powerful, but hard to productionize
- Needs actual DevOps team

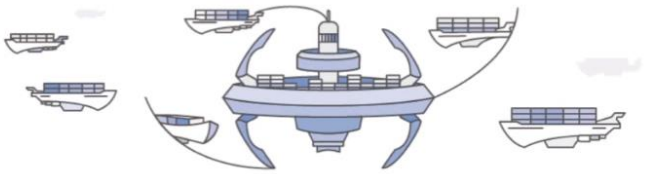
kubernetes by Google

- GCE first-class, everything else second-class

coursera

Amazon ECS

- Low-to-no maintenance solution
- Integrated with AWS infrastructure
- Easy to understand and program for



coursera

However...

- No scheduled tasks
- No fine-grained monitoring of tasks
- No retries / delays when cluster out of resources
- Does not integrate well with our

coursera

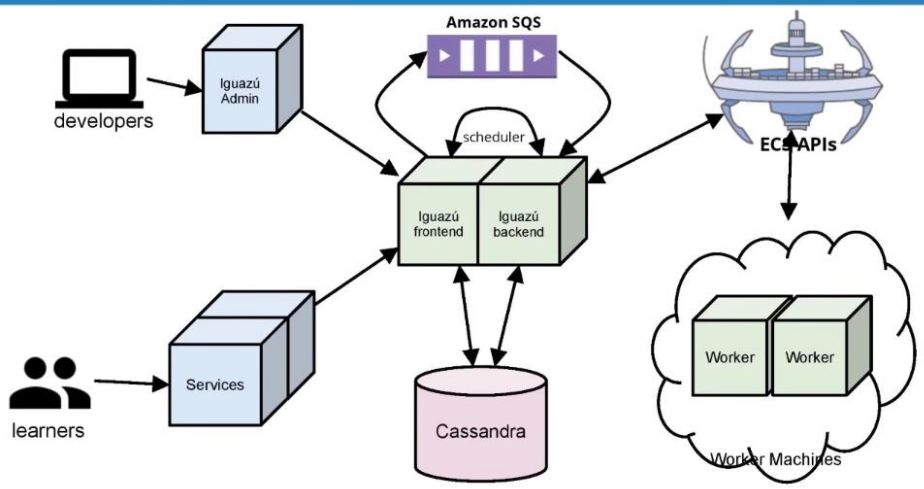
Iguazú -- Batch Management for ECS

- Named for Iguazú Falls
  - World's largest waterfall
- Batch Task Scheduler
  - Immediately
  - Deferred (run once at X time)
  - Scheduled recurring (cron-like)
- Programmatically accessible

Iguazú Semantic Guarantees

- **At most once** execution for all jobs
- Jobs will be provided with **at least** the **CPU** and **RAM** they requested
- Scheduler may elect to skip execution of some scheduled jobs under adverse conditions

Iguazú Architecture



coursera

Iguazú Design

- Frontend + Scheduler
  - Generates requests (either via API calls or internally from the scheduler)
  - Puts new requests in SQS queues
  - Handles requests for status from other services
- Backend
  - Attempts to run tasks via ECS API
    - Failure (e.g. lack of resources) means task goes back into queue to try again later
  - Keeps track of task status and updates Cassandra

coursera

Iguazú Admin User Interface

IGUAZÚ

Home

Jobs

Invocations

Scheduled Jobs

Iguazú Scheduled Jobs

Filter Scheduled Jobs

ID	Cron Expression	Next Invocation	Family	Job Name	Actions
319au6PDnnDwiKyP	0 15,45 * 1/1 * ? *	Today at 4:15 PM PDT	peerreviewgradereadyemail	peerreviewgradereadyemail	
ZtZny5H4OzocOEuN	0 59 23 1/1 * ? *	Today at 4:59 PM PDT	translationpipelineupdatesubtitlelanguages	translationpipelineupdatesubtitlelanguages	
pKMkmTpS8oVibYW1	0 0 20 1/1 * ? *	Tomorrow at 1:00 PM PDT	translationpipelinedownload	translationpipelinedownload	
9m9nRGN1JgjebbVw	0 0 7 1/1 * ? *	Tomorrow at 12:00 AM PDT	translationpipelineupload	translationpipelineupload	
yVCDUTGps3huOIZF	0 0 7 1/1 * ? *	Tomorrow at 12:00 AM PDT	mega	mega	
VYrHEGNFfHLQiAGL	0 30 17 1/1 * ? *	Tomorrow at 10:30 AM PDT	phoenixsessionemail	phoenixsessionemail	
gd7xlyxxcgelyjSp	0 0 18 ? * MON-FRI *	Tomorrow at 11:00 AM PDT	ondemandvideoreport	ondemandvideoreport	
neCafzkgCv1cCNTq	0 0 19 1/1 * ? *	Tomorrow at 12:00 PM PDT	searchindexupdater	searchindexupdater	
EdMQbhGOTCT77GcQ	0 0 12 1/1 * ? *	Tomorrow at 5:00 AM PDT	fullcoursetranslationupload	fullcoursetranslationupload	

FRANK CHEN
FRANKCHN@COURSERA.ORG

Iguazú Admin User Interface

IGUAZÚ

Home

Jobs

Invocations

Scheduled Jobs

E5fpbocaQOWlOm1qFAJHzg

Family	peerreviewmetrics
Invocation Time	July 7th 2015, 5:04:59 pm PDT
Requested Invocation Time	N/A
Status	SUCCESSFUL
ARN	
Command	

Filter Invocations

Invocation Time	Requested Invocation Time	Status	Actions
July 7th 2015, 5:04:59 pm PDT	N/A	SUCCESSFUL	
July 7th 2015, 5:09:59 pm PDT	N/A	SUCCESSFUL	
July 7th 2015, 5:14:59 pm PDT	N/A	SUCCESSFUL	
July 7th 2015, 5:14:59 pm PDT	N/A	SUCCESSFUL	
July 7th 2015, 5:14:59 pm PDT	N/A	SUCCESSFUL	
July 7th 2015, 5:14:59 pm PDT	N/A	SUCCESSFUL	
July 7th 2015, 5:19:59 pm PDT	N/A	SUCCESSFUL	
July 7th 2015, 5:29:59 pm PDT	N/A	SUCCESSFUL	
July 7th 2015, 5:29:59 pm PDT	N/A	SUCCESSFUL	

FRANK CHEN
FRANKCHN@COURSERA.ORG

Developing Iguazú Tasks

```
class Job extends AbstractJob with StrictLogging {
  override val reservedCpu = 1024
  override val reservedMemory = 1024

  def run(parameters: JsValue) = {
    logger.info("I am running my Job!")
    expensiveComputationHere()
  }
}
```

Running Tasks Locally

```
$ sbt
> project iguazuJobs
[info] Set current project to iguazu-jobs
> run sample sample.json
[info] Running org.coursera.iguazu.internal.IguazuJobsMain sample
sample.json
[info] 2015-07-08 13:31:52,368 INFO [o.c.i.j.s.Job] >>> I am running my job!
[success] Total time: 5 s, completed Jul 8, 2015 1:31:58 PM
>
```

Running Tasks from Other Services

```
// invoking a job with one command
// from another service via Naptime REST framework

val invocationId = IguazuJobInvocationClient
  .create(IguazuJobInvocationRequest(
    family = "mailer",
    jobName = "recommendationsEmail",
    parameters = emailParams))
```

Deploying Tasks

Easy Deployment

1. Merge into master. Done!

Jenkins Build Steps:

1. Builds zip package from master
2. Prepares Docker image
3. Pushes docker image into docker registry
4. Registers updated tasks with ECS APIs

Logs

- Logs are in /var/lib/docker/containers/*
- Upload into log analysis service (Sumologic for us)
- Wrapper prints out task name and task ID at the start for easy searching

Metrics

- Using third-party metrics collector (Datadog)
- Metrics for both tasks and container instances
- So long as can talk to Internet,
  things will work out pretty well

Programming Assignments

Programming Assignments

Programming Assignment: 编程作业

Instructions My submission Discussions

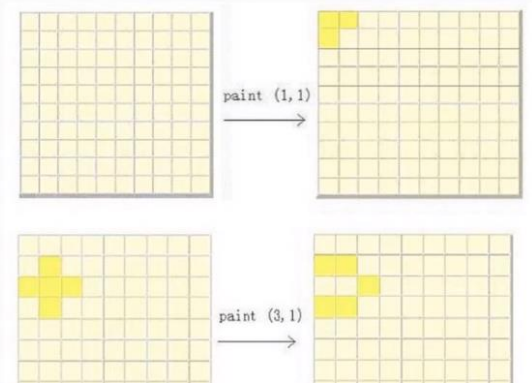
编程题 # 1： 画家问题

来源: POJ (Coursera声明：在POJ上完成的习题将不会计入Coursera的最后成绩。)

注意： 总时间限制: **1000ms** 内存限制: **65536kB**

描述

有一个正方形的墙，由N*N个正方形的砖组成，其中一些砖是白色的，另外一些砖是黄色的。Bob是个画家，想把全部的砖都涂成黄色。但他的画笔不好使。当他用画笔涂画第(i, j)个位置的砖时，位置(i-1, j)、(i+1, j)、(i, j-1)、(i, j+1)上的砖都会改变颜色。请你帮助Bob计算出最少需要涂画多少块砖，才能使所有砖的颜色都变成黄色。



How to submit

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

Programming Assignment: 编程作业

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.

编程题 # 1： 画家问题 50 points	source.cpp
编程题 # 2： 拨钟问题 50 points	source.cpp

Submit

Your Submissions

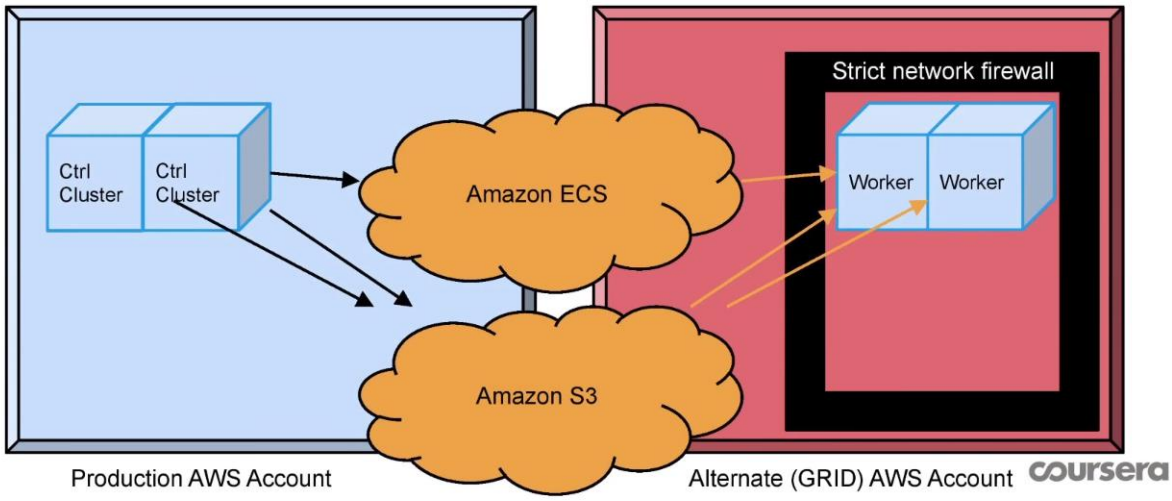
Graded submissions will appear here

Grading Prog. Assignments

- Special case of batch processing
- Near real-time feedback
  - <30 seconds for fast graders

- Compiling and running ***untrusted code***
- Infrastructure security huge concern
  - Minimize exfiltration of info (e.g. test cases)
  - Avoid turning into bitcoin miners or DDoS attack

GRading Inside Docker: Architecture



coursera

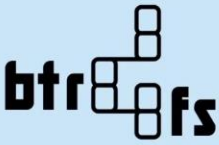
GRID: Defense in depth

Network

Completely separate AWS account

Network ACLs & Routing Tables

Host



Docker / Other

Custom cleaning of container images

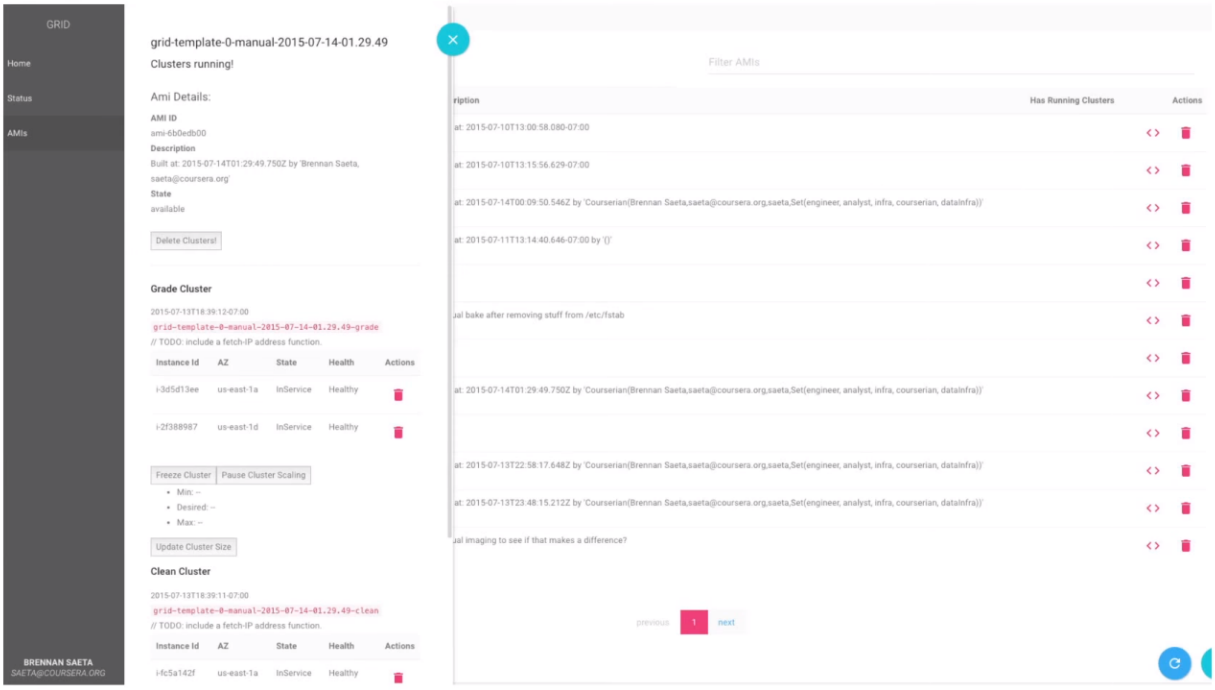
+ additional mitigation techniques and defenses

coursera

Modifying the ECS Agent

- Coursera has 2 simple forks of ECS agent
  - Allow privileged docker-in-docker access for the “cleaning” agent
  - Disable networking and disk writes for untrusted code in the “grading” agent

- Check it out at: github.com/coursera/amazon-ecs-agent



This is a screenshot of our admin interface

Usage

- Iguazú
  - **38 tasks written** since launch in April
  - **24 scheduled tasks**
  - **>1000 invocations** per day
- Grid
  - Pre-production at this time (launching in weeks)
  - Dozens of graders already written by multiple instructional teams!

Future Improvements

- True Autoscaling
  - Scaling up is easy, scaling down not so much
- Task prioritization (multiple queues)
- Simulate memory and CPU limits in dev modes

Lessons Learned / Docker war stories

- Docker instability fun
  - Container format changing between 1.0 and 1.5.
- btrfs wedging on older kernels
  - Default 14.04 kernel not new enough!
- Disk usage
  - Docker-in-docker can't clean up after itself (yet).