Faculty Institute 2020 Continuous Integration and Delivery in Student Projects

Dr. Mark Hills
Associate Professor, Department of Computer Science, East Carolina University

Google Cloud

Mark Hills

Associate Professor
Department of Computer Science
East Carolina University

After working professionally as a software engineer, I went back to school and got my PhD. I've been at ECU since 2013, teaching courses related to software development and software engineering.

Google Cloud

Motivation

- Concepts related to DevOps are becoming increasingly prevalent in practice
- This includes continuous integration, containers, build automation, and cloud deployment

- How can we prepare our students for this?
- What backgrounds do they need to be able to learn these skills?
- Are there activities we can use to give them hands-on experience?
- If so, what are some examples of these activities?

A quick aside

- ITiCSE Working Group 7: Cloud Computing Curriculum: Developing Exemplar Modules for General Course Inclusion
- See https://iticse.acm.org/working-group-details/ for details

- I have been involved in this working group this year, but this is work from before joining
- I am not speaking for the working group, this is not a presentation of the work of WG-7

- Parts of this are also described in a CSEE&T 2020 short paper I authored

Introduction: What will we cover?

- A series of hands-on activities related to continuous integration, build automation, and continuous delivery
- Details of a student project that uses these techniques
- Some lessons learned

Introduction: Who is this for?

- The main audience for this presentation is faculty teaching undergraduate and graduate courses in computer science and software engineering (more about the course context coming up!)
- The secondary audience is interested faculty or professionals that would like to learn some of this material themselves and incorporate it into their own work

Overview: Computer Science at ECU

- Two undergraduate programs: BS in Computer Science, BS in Software Engineering (new!)
- Three graduate programs: MS in Computer Science, MS in Software Engineering, MS in Data Science

- Graduate programs attract a range of students
  - Fresh undergrads
  - Working professionals
  - Career switchers

- BS in CS has two SE courses (one regular course, one project/capstone course)

Zooming in: Undergrad context

- Software Engineering II: Undergrad project/capstone course

- Students work in teams of 5 or 6
- Projects based on student interest, some from outside, some proposed by students
- Some projects use web or mobile, use cloud credits to give students experience
- Note: students are encouraged, but not required, to use cloud

Zooming in: Grad context

- Software Construction: Grad software development course

- Students generally work individually
- Focus is on scaling up development skills for larger software systems
- Important concepts: abstraction, code comprehension/understanding, professional practice
- Lots of non-cloud skills: IDEs, version control (Git), build automation, unit testing, test mocks, code coverage, lightweight static analysis (SpotBugs/PMD)

- Originally, no focus on cloud

Activity origins: First attempt

- Goal: DevOps concepts are becoming widely used in practice, students should have some experience with them!

- First attempt: use existing “getting started” docs, online example, some online readings
- Goal was to get students to try things, not graded beyond participation
- This...didn’t go so well

Activity origins: First attempt problems

- Sometimes, technologies change faster than docs (minor issue)
- Many docs written for professionals with richer backgrounds (major issue)
- Students could not differentiate fundamentals from examples (major issue)
- Students would get lost, not know how to get back on track (major issue)

Activity origins: Second attempt

- Started over from scratch, created 4 activities

- Activity 1: Continuous Integration
- Activity 2: Docker
- Activity 3: Kubernetes
- Activity 4: Continuous Delivery

- Mixture of instructor-provided material, videos (LinkedIn Learning), vendor docs
- Ready-made “starters” for each step
- Participation credit, not formally graded (low stress)

The Hello-izer

- Activities use a simple Java Spring app, The Hello-izer!
- RESTful Web Service with one endpoint
- Only knows how to say “Hello”!

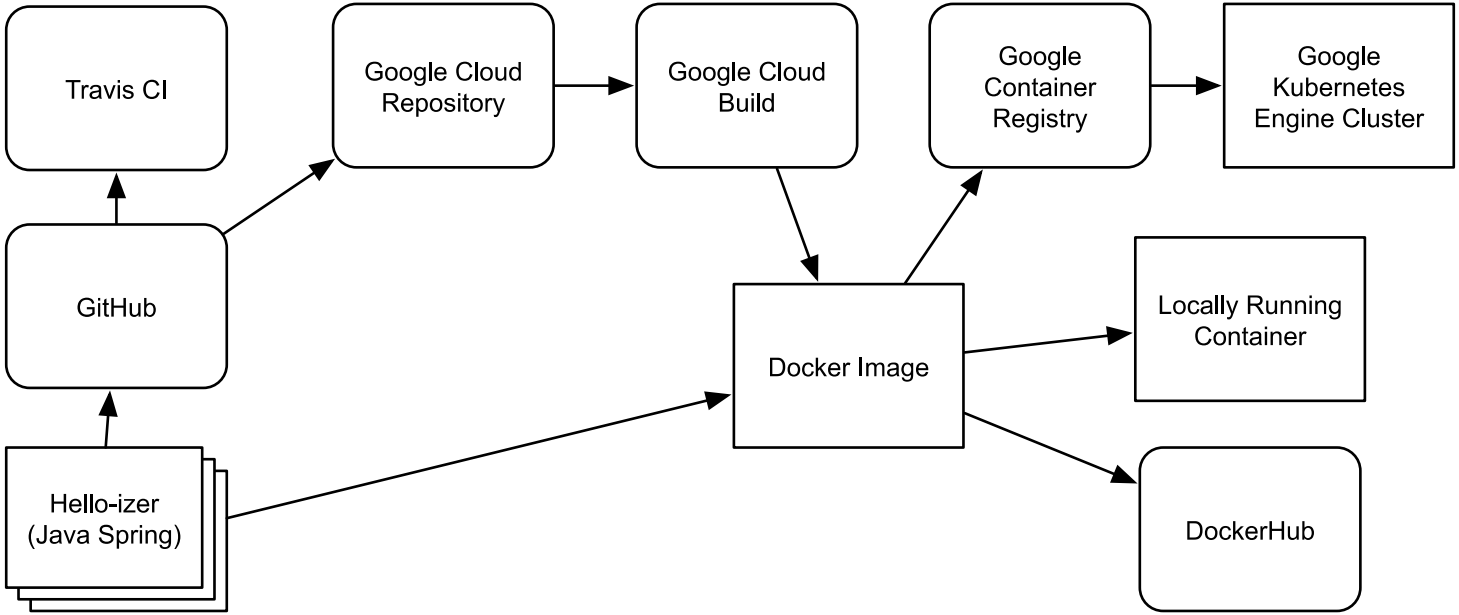
$ curl http://localhost:8080/hello

{"messageId":"3d071f7e-bd26-4b26-b77a-891ec1449da0","message":"Hello, anonymous person it is Fri Sep 04 16:06:52 EDT 2020","messageDate":"2020-09-04T20:06:52.749+0000"}

$ curl http://localhost:8080/hello\?name\=Mark

{"messageId":"284172ed-c4c1-4ced-bdce-c6f6e422b7f4","message":"Hello, Mark it is Fri Sep 04 16:06:58 EDT 2020","messageDate":"2020-09-04T20:06:58.199+0000"}

Activity overview: Moving parts

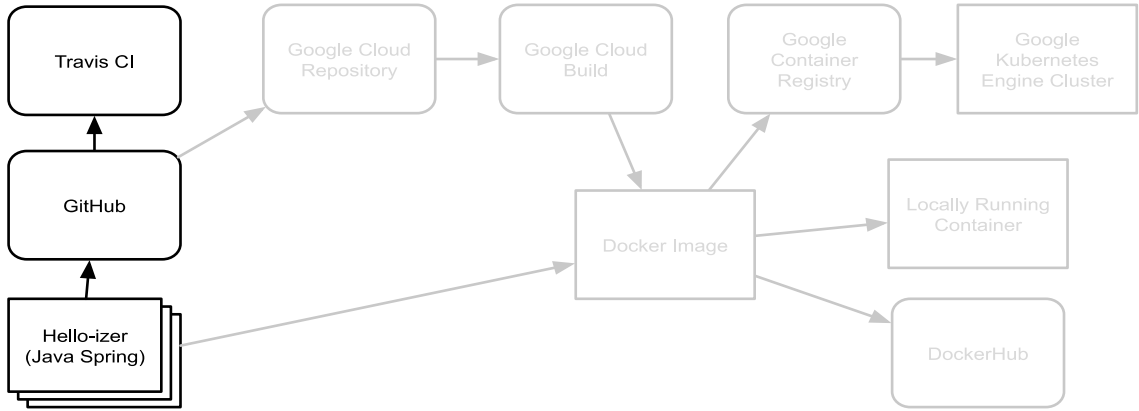


Google Cloud

Activity 1: Continuous integration

- Overall goal: students enable CI, see it in action

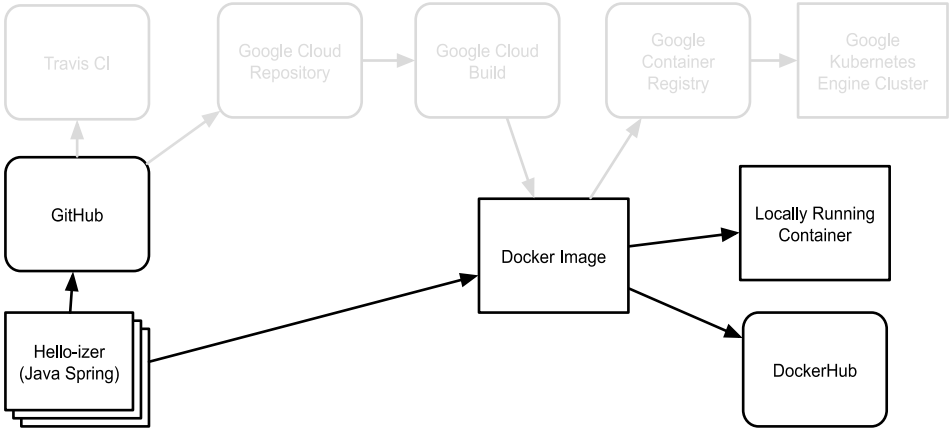
- Students fork Hello-izer repo
- Need to enable Travis-CI integration
- First push to GitHub will trigger Build with failing test
- Students fix code, push changes
- Now, all tests pass



Activity 2: Docker

- Overall goal: students learn how to build and share a Docker image, automate the build, run and test the container

- Students start with Activity 1 code
- Need to create Dockerfile to create the image
- Test the image to see if it can be used to make a working container
- Automate container build process
- Push results to DockerHub and GitHub

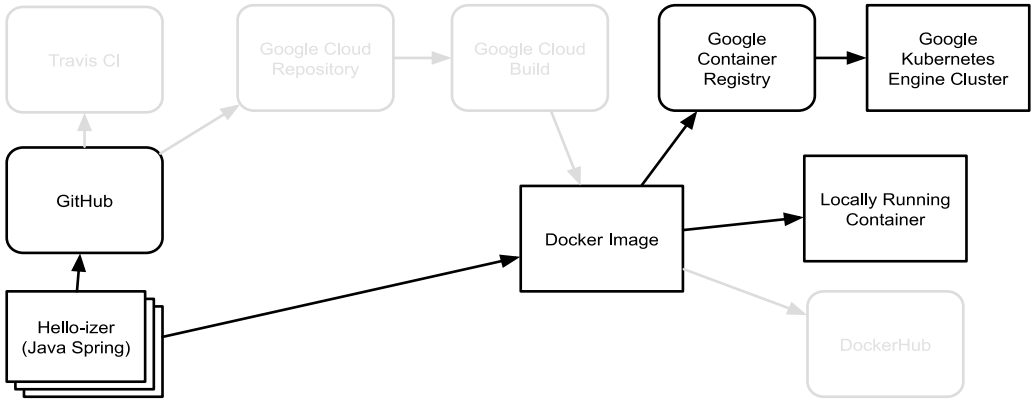


Google Cloud

Activity 3: Kubernetes

- Overall goal: students learn how to deploy a Docker image to a Kubernetes cluster

- Students start with image from prior Activity or from instructor
- Set up project on Google Cloud
- Use SDK to push image to the Container Registry
- Create a Kubernetes cluster
- Deploy image to the cluster
- Make available using load balancer

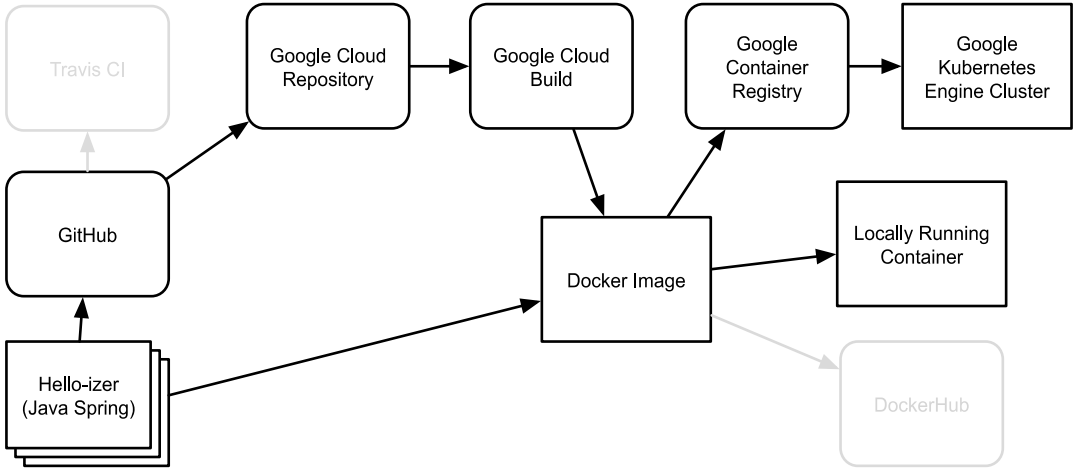


Google Cloud

Activity 4: Continuous delivery

- Overall goal: students put this all together to deploy changes to their cluster

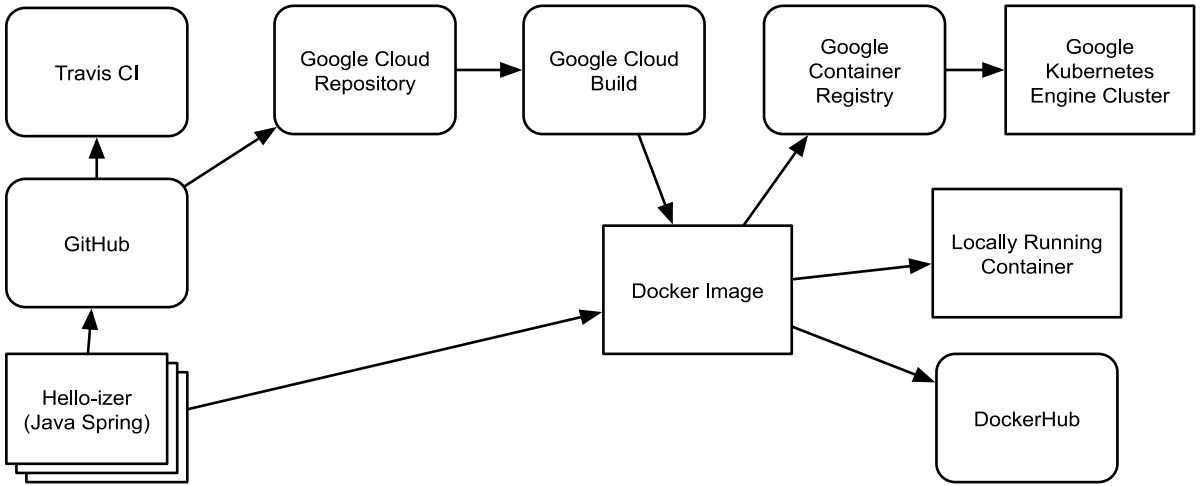
- Students create a Cloud Source Repository that syncs with GitHub
- Cloud Build enabled and configured
- Trigger added to trigger build when repository changes
- Check build by deploying to Kubernetes, then extend build to automate deployment



Google Cloud

Activity summary

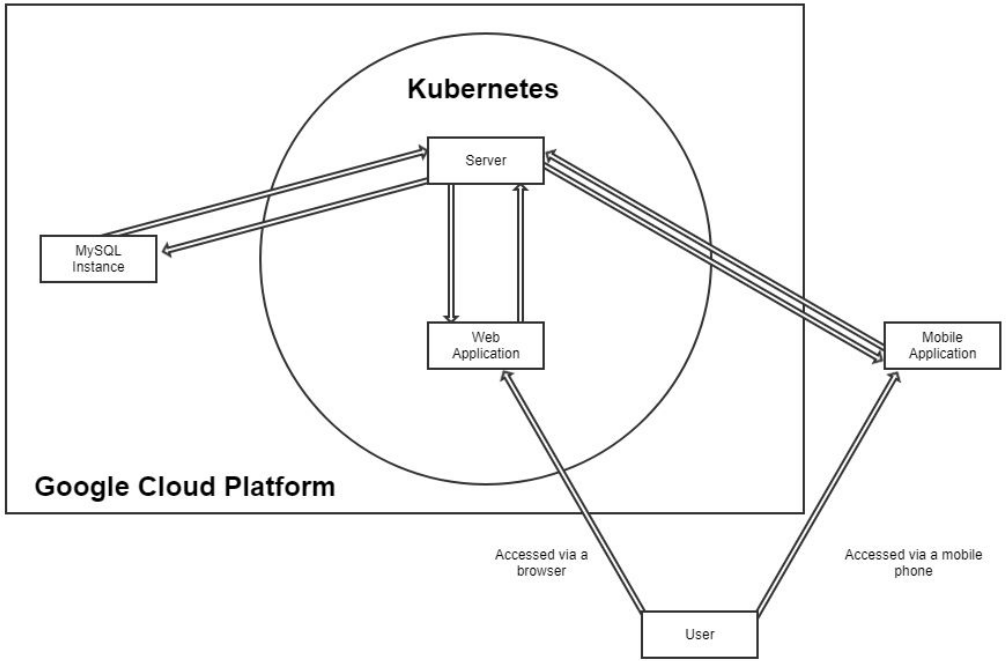
- By the end, changes to repository trigger test and creation of new Docker image
- Kubernetes automatically updated using newest changes committed to GitHub
- Students can change API output and quickly see deployed changes



Google Cloud

An example student project

- Note: image from their documentation
- Secure messaging app named Spark
- API and web server deployed to Google Cloud whenever GitLab is updated
- Uses Docker and Kubernetes



Google Cloud

Student project: deployment details

```
deploy:
  image: google/cloud-sdk:latest
  stage: deploy
  variables:
    DOCKER_IMAGE_TAG: theconartist/spark-web-application:latest
  script:
    # Authenticate with GKE
    - apt-get --assume-yes install gettext
    - echo "$SACK" > key.json
    - gcloud auth activate-service-account --key-file=key.json
    - gcloud config set project spark-266522
    - gcloud config set container/cluster spark-web-cluster
    - gcloud config set compute/zone us-east1-b
    - gcloud container clusters get-credentials spark-web-cluster --zone us-east1-b
    - sed -i "s/<VERSION>/${CI_COMMIT_SHORT_SHA}/g" deployment.yaml
    - cat deployment.yaml | envsubst | kubectl apply -f -
```

Student project: deployment details

```
spec:
  containers:
  - name: spark
    image: theconartist/spark-web-application:<VERSION>
    env:
      - name: "PORT"
        value: "19006"
```

Lessons learned

- Command line skills are important, but often missing
- Vendor documentation is important, but often missing context students need
- Kubernetes is tricky for students (and sometimes for faculty!)
- Don’t use latest with Kubernetes for Docker images, it doesn’t do what you think it does!
- Be ready to work with your students, it’s important they learn this material, but they will get stuck on parts of it
- Be patient with yourself, it’s hard to keep up with all this stuff!

Following up & next steps

- Working on converting activities to CodeLabs format
- Feel free to contact me if you want to discuss these! I'm at: hillsma@ecu.edu

Thank you.