OPENedX 2018

Running and Developing Open edX in Devstack – Docker Edition

Jeremy Bowman

MONTREAL, QUEBEC, CANADA

Who Am I?

Jeremy Bowman

Principal Software Engineer at edX

Tools Team Tech Lead

Docker Devstack contributor

Tentative Workshop Schedule

- Offline devstack installer
- Explain what devstack is (and isn’t)
- Explain the installation process
- Demo!
- Audience choice:
    - Q&A
    - Tackle installation or usage issues in small groups
    - Deeper dive into particular topics

System Requirements

- Linux or macOS
- Docker 17.06+ CE - https://www.docker.com/community-edition
- make
- Linux: docker-compose 1.9.0+
- 2+ CPU cores, 6+ GB memory, 25+ GB storage (32 for offline installer)

Windows support should be feasible (just not attempted yet)

Open edX Devstack Offline Installation

1. Copy devstack_snapshot from one of the flash drives to your computer
2. Unmount and return the flash drive
3. Follow the instructions in devstack_snapshot/README.txt
4. Ask if you have any questions!

What is Devstack?

- The default development environment for Open edX code improvements
- Not intended for production use
- A collection of Docker containers
- No longer based on Vagrant

Docker – Development vs. Production

- Devstack's Docker images are optimized for development
- Include lots of libraries and tools that you wouldn't want in production
- Source code mounted from host OS, not internal to container
- Designed to work like Ansible-configured virtual machines (the current recommended production environment)
- Docker images for production would be built very differently
- Experimental production images have been created by the community
- Work to reduce the differences is being scheduled

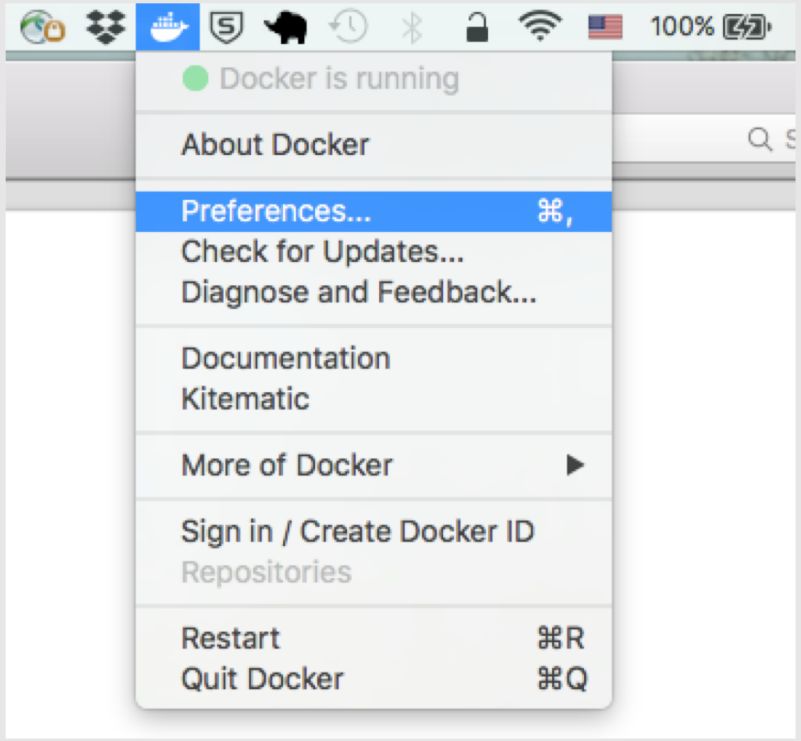
Changes From Vagrant Devstack

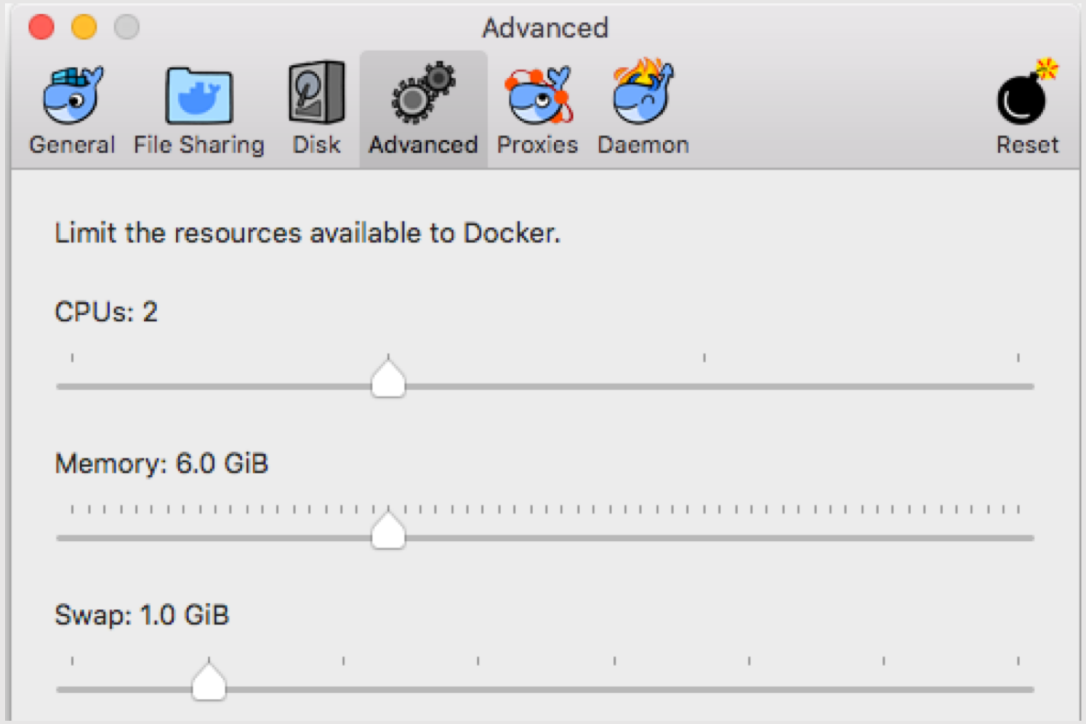
- No VM needed in Linux!
- VM is easier to manage in macOS
- New services: Analytics Data API, Analytics Pipeline, Credentials, XQueue
- Removed service: Programs
- Service isolation similar to a production configuration
- All Open edX services running from source on host OS

Hawthorn Installation Options

- Docker Devstack
- Native
- ~~Fullstack~~
- Community Docker images

Docker for Mac Settings





Clone the devstack Repository

URL: https://github.com/edx/devstack/

$ git clone https://github.com/edx/devstack.git

OR

$ git clone git@github.com:edx/devstack.git

Makefile

```
C02QX0W0FVH6:devstack jeremybowman$ make
Please use `make <target>' where <target> is one of
  analytics-pipeline-shell  Run a shell on the Analytics pipeline container
  backup                    Write all data volumes to the host.
  build-courses             NOTE: marketing course creation is not available for those outside edX
  check-memory              Check if enough memory has been allocated to Docker
  clean-marketing-sync      Remove the docker-sync containers for all services (including the marketing site)
  create-test-course        NOTE: marketing course creation is not available for those outside edX
  credentials-shell         Run a shell on the credentials container
  credentials-static        Rebuild static assets for the credentials container
  credentials-update-db     Run migrations for the credentials container
  destroy                   Remove all devstack-related containers, networks, and volumes
  dev.clone                 Clone service repos to the parent directory
  dev.provision.run         Provision all services with local mounted directories
  dev.provision             Provision dev environment with all services stopped
  dev.repo.reset            Attempts to reset the local repo checkouts to the master working state
  dev.reset                 Attempts to reset the local devstack to a the master working state
  dev.status                Prints the status of all git repositories
  dev.sync.daemon.start     Start the docker-sycn daemon
  dev.sync.provision        Provision with docker-sync enabled
  dev.sync.requirements     Install requirements
  dev.sync.up               Bring up all services with docker-sync enabled
  dev.up.all                Bring up all services with host volumes, including watchers
  dev.up.analytics_pipeline Bring up analytics pipeline
```

Clone Service Repositories

```
$ make dev.clone
$ ls
course-discovery            edx-analytics-pipeline
credentials                 edx-e2e-tests
cs_comments_service         edx-platform
devstack                    xqueue
ecommerce
```

Provision Data Volumes

$ make dev.provision

(lots of Ansible output and 20+ minutes of time)

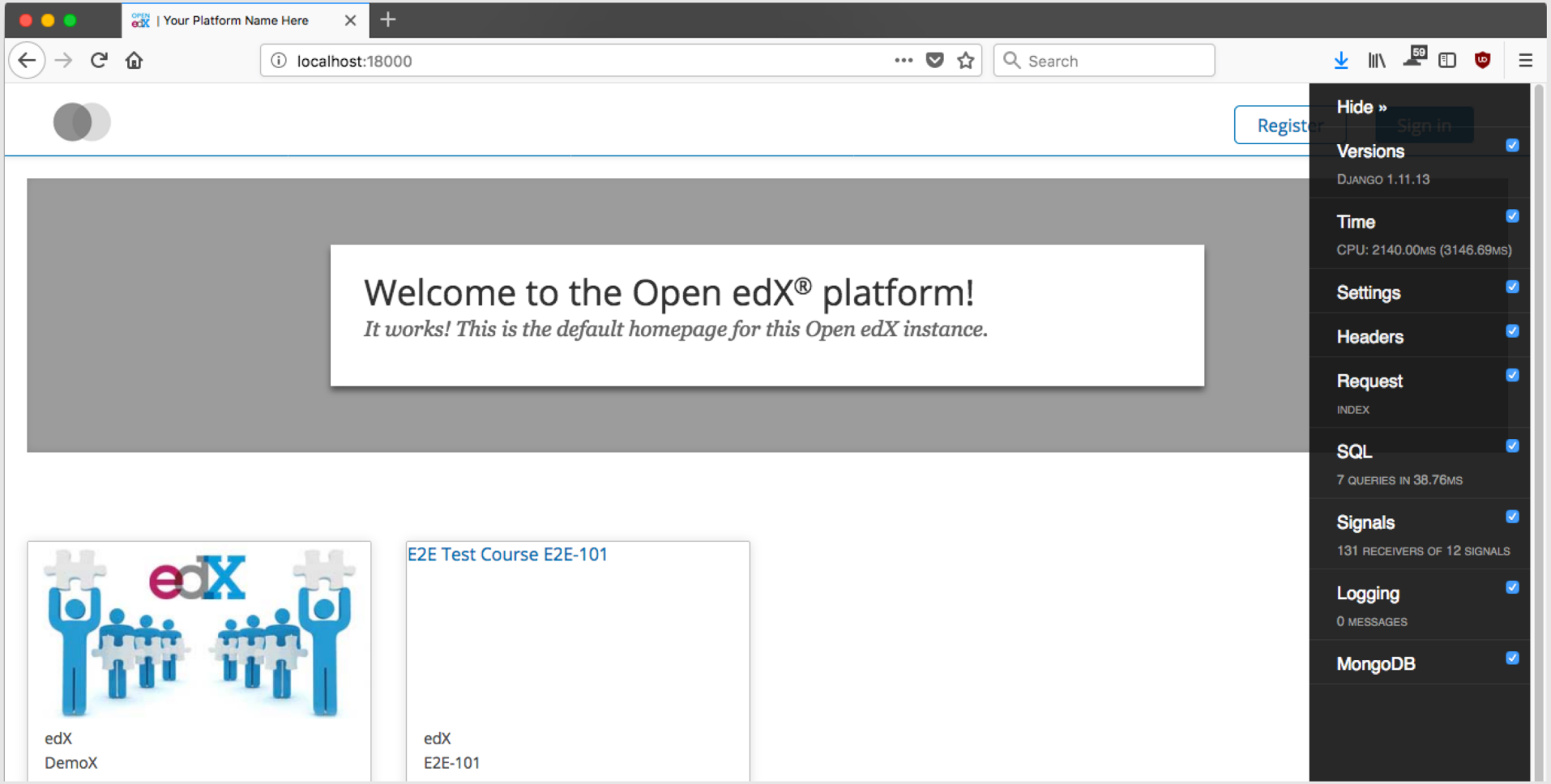
- Updates Python and Node.js dependencies to match source on host
- Creates and populates databases
- Collects static assets

Note: This wipes out any existing database content!

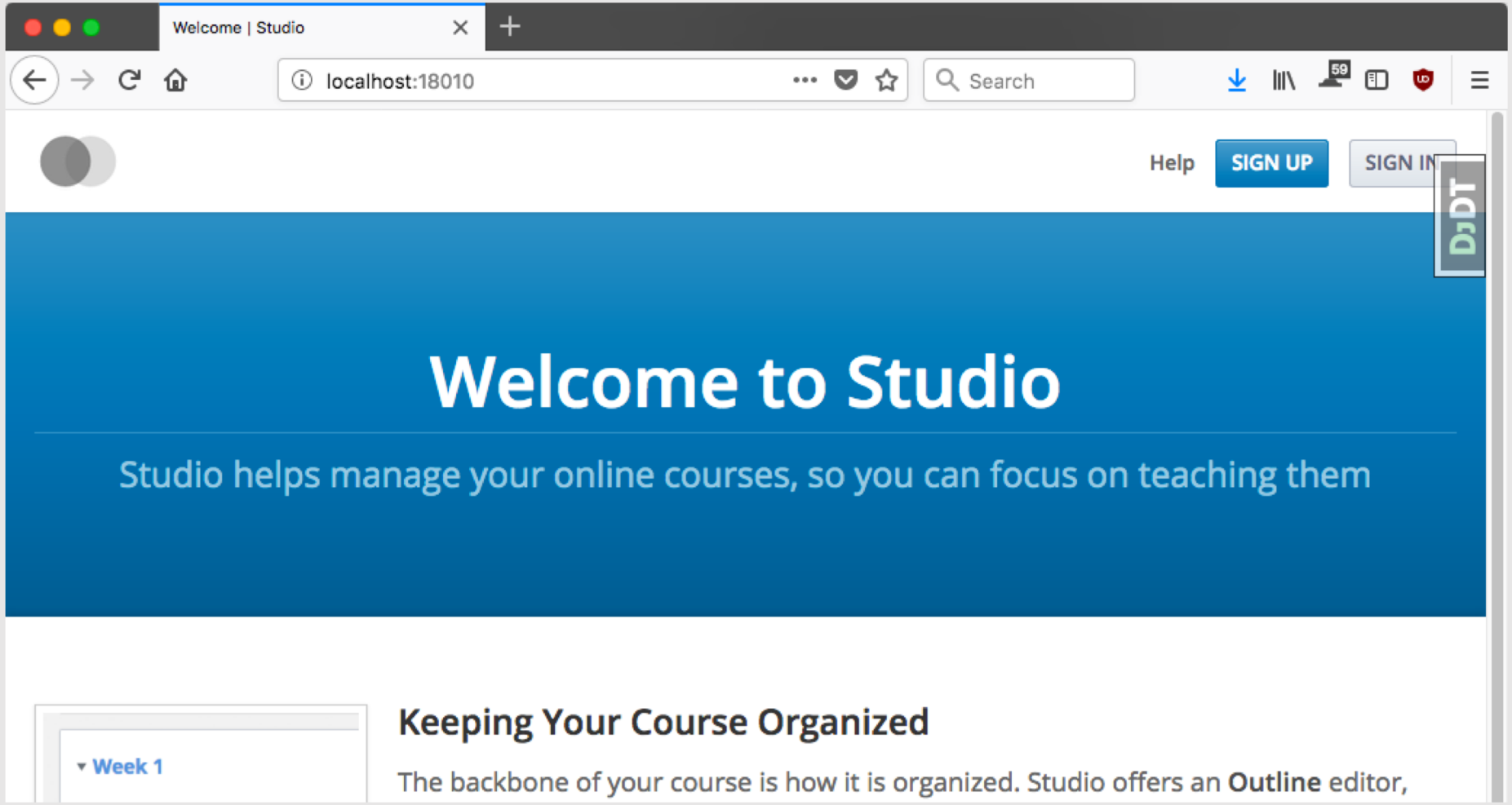
Start the Services

```
C02QX0W0FVH6:devstack jeremybowman$ make dev.up
docker-compose -f docker-compose.yml -f docker-compose-host.yml up -d
Starting edx.devstack.chrome         ... done
Starting edx.devstack.devpi          ... done
Starting edx.devstack.mysql          ... done
Starting edx.devstack.elasticsearch ... done
Starting edx.devstack.mongo          ... done
Starting edx.devstack.memcached      ... done
Starting edx.devstack.firefox        ... done
Starting edx.devstack.ecommerce      ... done
Starting edx.devstack.credentials    ... done
Starting edx.devstack.discovery      ... done
Starting edx.devstack.forum          ... done
Starting edx.devstack.lms            ... done
Starting edx.devstack.studio         ... done
./programs/provision.sh cache >/dev/null
```

LMS on Port 18000



Studio on Port 18010



Pre-Populated Users and Demo Course

Username	Email	Notes
edx	edx@example.com	superuser
audit	audit@example.com	
honor	honor@example.com	
staff	staff@example.com	
verified	verified@example.com	

Default password for all these users is “edx”

Read the README

The devstack repo's README is packed with useful information:

- Service URLs
- Shell access
- Inspecting logs
- Running tests
- Keeping devstack up to date
- Debugging with PDB or Pycharm
- Troubleshooting common problems

Audience Participation Time!

- Questions?
- Anybody need help with their devstack setup?
- What do people most want to learn next?

Shell Access

$ make lms-shell

- Also studio-shell, mongo-shell, mysql-shell, …
- Activates virtualenv and nodeenv (if any)
- Try “paver help” and “make help”

Running Tests

- paver
- pytest
- tox
- Chrome and Firefox available for browser automation tests
- Those browsers can be accessed via VNC
- For edx-platform, see docs/testing.rst
- make e2e-tests

Logs

Convenience targets for typical docker-compose log display commands:

$ make logs

$ make studio-logs

…

Don’t need to go hunting for log files

Also useful: make stats

Getting new Docker Images

$ make pull

- Automatically released on merged changes
- Not small
- Remember to also “git pull” in relevant repos, including devstack

Exercise Suggestions

- Start the demo course
- Make a change to the demo course content via Studio and verify it in LMS
- Make an HTML, CSS, or JS change and see it applied in the running service
- Run the unit tests for lms.djangoapps.lti_provider (or another package)
- Start a single bok-choy test module and watch it run via VNC
- Find and fix a pylint warning
- Browse the Discovery or LMS APIs and try a few calls
- See if it works on Windows…

Thank You!

Open edX Slack channel: #docker

Source code: https://github.com/edx/devstack/