CSE 403

Deployment (and more...)

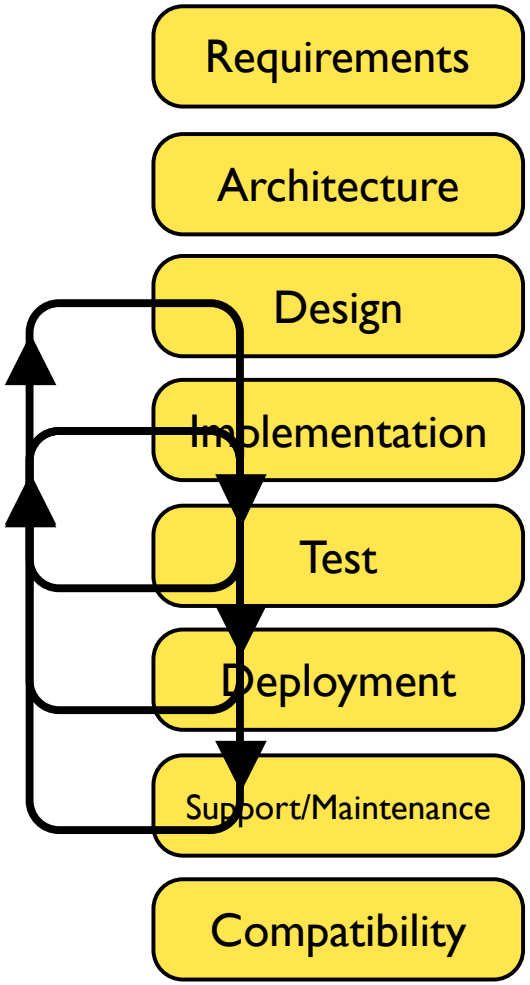
Deployment

Software deployment is all of the activities that make a software system available for use.

The type dictates the what gets deployed and how

- Consumer web apps, enterprise, embedded systems, native phone apps, Curiosity: They are all different
- Yet, general principles remain similar

Lifecycle Tasks



What

Big Picture

Details (but...)

Code

Make sure it works

What the customer sees

Keeping it working

Old versions work with new

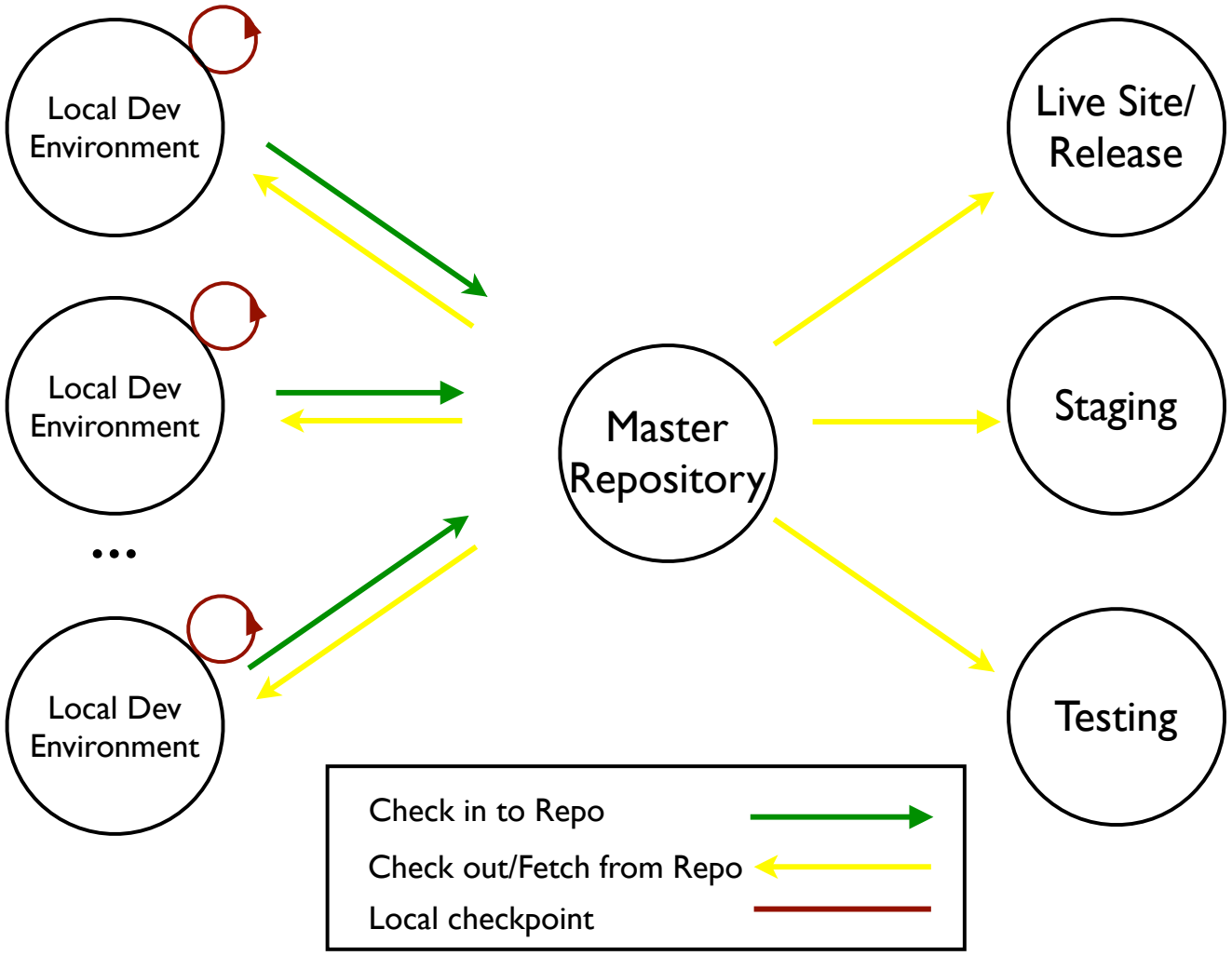
Software Environments

- Development environment
- Testing/Integration Environment
- Customer/Live Environment

Software Environments

- Development environment
- Testing/Integration Environment
- Staging Environment
- Build Environment
- Customer/Live Environment

Version control system typically ties it all together



Development Environment

- Optimized for developers to implement (and unit test) software
- Close to a production environment but not exact
- Standardization across all developers important(?)
- Portable(?)
- Easy to spin up (and spin down)
- Where do you get test data/databases?
- Includes access to VCS, bug tracking, wiki...

Live Environment (for [small] web applications)

- Clean separation from all other environments
- What gets incrementally installed?
- How does it get installed?
- Tools for installation
- Tools for monitoring
- Caveat: Large scale deployments might be very different

Testing Environment

- An environment for testing outside of the development environment
  - Regression tests, verify human interactions, multi-platform (browsers, devices)
- Clean separation from all other environments
- Nightly bring ups? On demand? Continuous?
- How does it get installed?
- Reasonably close live environment
- Caveat: Large scale deployments might be very different

Staging Environment

- Very close replica of live environment
- Only very small deltas
- Including installation procedures, monitoring, ...
- Backup,operations, scalability, reliability may be different
- Not needed in small organizations? (testing/staging are merged)
- Not possible in big organizations?

Build environment

- Intermediate step from source code to runnable software
- Not always necessary or visible
- Developers can usually build in their dev environments
- Compile, linking, and packaging for deploying to testing, staging, live environments
- Careful understanding of the tools and versions used to build and run

Show me the data

- Live data on the live site
- Test data on the staging site, test site
- Data for development
  - Shared database for all developers?
  - Individual db's for each developer
  - Developer RDBMS same as live, staging, test?

A good hack...

- For small/early/non-transactional web deployments
- Copy data over (dump the db) from the live site to the staging/testing sites on a nightly basis
- Gets usable data to staging and testing
- Backs up data from the live site
- Warning: hack is lossy

How to deploy

- “Bootstrap” (first time)
- Deployments
- Maintenance/Monitoring
- Provisioning/scale out

“Bootstrap”

- Don’t have to deploy from “ground zero” every time -- just the first time
- Operating system, database, patches, libraries, services (webserver?): stuff that really isn’t changing very often
- Can you “snapshot” this for future use?
- Related to “provisioning” -- a rapidly evolving topic

Deployment

- Pre-condition: Satisfied with all changes/additions going into a release
- Precondition: Passed all testing requirements (including staging testing, if staging)
- How do you get the code onto the live site?
- If there is a build process, the output of the build is the input to the deployment process

Deployment (small scale systems)

- Can you deploy "hot?"
- Are there data (schema) changes?
- Are there infrastructure changes?
- Do you need to quiesce the system?
- What is the mechanism to get the new code installed?
- How do you bring up the system?
- Who do you need to tell?
- What happens if the deployment fails?
- Can you "roll back" if there is a major problem?

Installing new code

- Just pull (or push from the repository)...Really?
  - The Heroku way?
  - I've done this with SVN
- "Shell" hacking involved?
- Tools: Chef, Fabric, Tddium?
- Can you do this in one step?
  - Existence proof: %fab production deploy

Provisioning

- Bringing up new instances of your live environment
- Scaling
- Adding new servers/machines/VMs
- Vagrant, Puppet

Pulling it all together

Release life cycle (from coding to deployment)

- Merrily hacking along
- Check in code as feature get implement
- Regression testing (more on this later)
- Nightly push to testing?
- Don't break the "build" (nightly tests)
- Code freeze/feature complete

Release life cycle (from coding to deployment)

- Only checkins to fix bugs
- Feature ("monkey") testing on test site
- Check list of features being completed, what's outstanding
- Reviewing bug list, fix only show stoppers (more on this later this week)

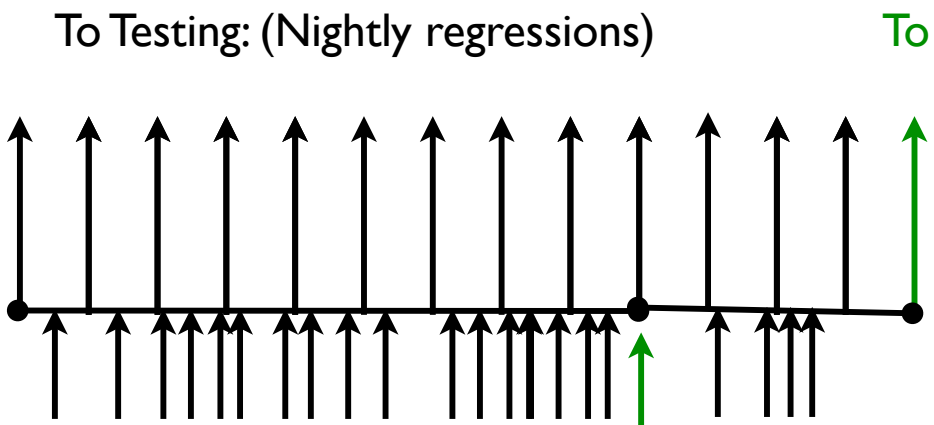
Release life cycle (from coding to deployment)

- Test system good?
- Quiesce the live site?
- Deploy
- “Test” the live site

Release life cycle (from coding to deployment)

- Not working?
- Recommendation: Can't turn back, make progress moving forward
- Monitor
- "Hot" fixes?
- Question: What day do you roll the live site?

Release time line



What if you are doing continuous releases?

- Check in changes from developer environment and to the live environment
- Regression testing in the development environment/test environment before pushing
- Works for smaller changes, but what about bigger, riskier changes?
- What about infrastructure changes?
- Changes with lots of dependencies?

Larger scale deployments

- Replication and redundancy -- how do you deal with this?
- How do you bring in new resources?
- How do you monitor and manage?
- Really big deployments: Entirely different game

Operations/ Maintainence

- Availability
- Reliability
- Scalability
- Performance
- Backup
- Monitoring (small scale: munin, Google webmaster, Google analytics, pingdom)

Who does deployments/operations?

- Big companies, operations/deployment is usually separate from an engineering organization
- Small companies oftentimes too
- I favor bringing it in close to engineering
  - Engineers should know what it takes to deploy and operate their software ("Feel the pain")
  - More efficient
  - Greater understanding

What about software product releases?

- In contrast to websites/hosted applications
- Assets come out of the build environment
- Attention to platforms, environments, and versions where the software will run
- Synchronization betweem development, build, test, and customer environments crucially important
- Binary vs. source distributions?
- Third party software; bundled or not
- Distribution? Over the web?
- Version management, documentation, support bigger issues

Take aways

- Deployment of software varies greatly depending on the type and size the product
- Basic principles remain the same
- There are different environments in the software lifecycle
- One step deployments should be your goal
- There are tools to help
- Large scale systems are different!