THE STATEFUL CONDITION: OR HOW I LEARNED TO STOP WORRYING AND EMBRACE THE CLOUD

Sara Jeanes
NET+ Program Management, Internet2
“Machine” Method

- On-Prem (Private Cloud)
- Burstable (Hybrid Cloud)
- Public Cloud
- "Stateless"
- Serverless
“Human” Method

1. Fully Manual (hopefully virtualized by now)
2. Fully Manual but monitored (reactive)
3. Manual Ops Deploy/Automated Recovery
4. Possibly manual failover
5. Automated Deploy (initial state)/Automated Recovery
6. Automated Deploy (changed state/update/delta)/Automated Recovery
7. Auto Code Commit (to Dev environment?)/Manual Deploy to Prod
8. Automated Code Commit and Test (to Test environment?)/Manual Deploy to Prod
9. Automated Code Deploy/Automated Deploy to Prod
10. (Nirvana) A/B testing of changes
Private Cloud - Current

- Virtualized (hopefully)
- Largely Manually Configured at the OS level
- Pets over Cattle
- Disconnect between Server config an Application config

- How to start setting a good beginning:
  - Change management practices: Code, Config, Lifecycle
  - Inventory
  - Beginning to sound like ITIL....
- What are the needs of your customer?
  - Not apps! Ask the workflow, data, desired outcome questions
Hybrid Cloud

- Burstable when needed but still data center centric
- Beginnings of automation (primarily for spinning up/spinning down)
- Builds understanding of existing technical debt
- Does my IT infrastructure have the maturity to contemplate this option?
- 12-18 month timeframe

- How to move here:
  - Understand the cycle – academic, dev/test
  - Scripting your deployment workflow
Public Cloud

- Scale!
- Web centric development, batching, large but async jobs
- 3-5 year timeframe
- 6-9 months for green field

- How to approach
  - Assessment of application’s compatibility – SDLC – does it make sense?
  - Culture: What layers of the stack do you manage?
  - Understand network and security consideration
Public Cloud (for Stateless[ness])

- All data persistence is stored separately from compute
- Resources/instances are provided as needed
- Destroy and Rebuild
- Blue/Green Deploys

- Approach
  - An evolution on Public Cloud: Advanced Public Cloud
  - A likely current state end goal for a commercial or home grown web app
  - See Containers Slide
What Does a Proper Microservices Software Architecture Look Like?

• Each microservice has a lean connection to every other microservice, usually through a RESTful API.
• Microservice boundaries are drawn around organizational capabilities — perhaps around particular development teams.
• Each microservice is deployed and updated independently of other microservices.

– http://thenewstack.io/genius-techie-doesnt-really-understand-cloud/
What about containers?

- Docker, and others, but mostly Docker
- An enabler of microservices/SOA/
- A growing ecosystem, even more robust than the VMware virtualization one
  - Orchestration
  - Revision Controls
  - Security and Packaging
  - Hosted environments
  - Network and Storage discoverability tools
Serverless[ness]

- Are you building a mobile app for your incoming freshman class? Try Serverless!
- Backend-as-a-Services
- Mobile app – the rise of the “thick client” (again)
- Web Layer - App Layer - Database becomes Client – Database

- Approach
  - Rapid prototyping (i.e. in two days)
  - Especially well suited for mobile development
One Specific Example

Open Source PaaS

Automated Infrastructure

Domain Specific Use Cases

- big data & batch
- data services: cache, database, messaging...
- infrastructure services: logging, monitoring...
- 12-factor apps & PaaS
- microservices (eg. Netflix OSS)
- CI/CD

Container Service

Domain Specific Use Cases

- Docker
- Kubernetes
- Netflix OSS
- Shippable
- Mesosphere
- Amazon Web Services

What is DevOps?

• First Public Sighting
  – Allspaw and Hammond at Flickr
  – Velocity 2009
  – 10 deploys a day
  – http://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr
• Culture. Culture. Culture.
• Dev teams and Ops teams talking to each other!
How do you DevOps?

• Decrease impact to customer of disruptions
  – Recoverability over Nines
  – SLA levels are expected
• Infrastructure and application as code
  – Joined at the hip
  – Same pipeline and testset
• Scrum is a great place to start
• Continuous Integration = iterate often
• Continuous Delivery = provide incremental value, constantly
What Next?

- Assess
- Try in production!
- Build Momentum
- Think iteratively
  - The stack is not precious
  - Serve the mission not the app/tool
- Always be learning (we are Higher Ed after all)
Resources

• The Phoenix Project. Gene Kim, Kevin Behr, and George Spafford. 2013.
• The New Stack (http://thenewstack.io)
  – “Your Lead Architect Doesn’t Really Understand Microservices”
  – “How the Enterprise Adopts the New Stack, or “I Said No, Dammit”
  – “The Post-Amazon Challenge and the New Stack Model” (spin warning)

• ECAR
  – http://www.educause.edu/library/resources/preparing-it-organization-cloud
  – http://www.educause.edu/library/resources/cloud-strategy-higher-education-building-common-solution
  – http://www.educause.edu/library/resources/tco-cloud-services-framework