Converting Technical Debt into Campus Assets

Paul Jamason and Ryan Rose
Academic Technology Services / IT Services
Background
Academic Technology Services: Functions and Services

- Provide technology services, support, and resources to the UC San Diego academic community, facilitating research, teaching and learning
- Learning Management Systems (canvas.ucsd.edu; online.ucsd.edu)
- Machine Learning Platform (datahub.ucsd.edu)
- Lecture Capture (podcast.ucsd.edu; mediaspace.ucsd.edu)
- EdTech Support and Consultation (edtech.ucsd.edu)
- Media Teaching Labs and Facilities (e.g., StudioU)
- Research IT Services
- Instructional Computing Services: Student and course setup processes
ATS Student and Course Setup Processes Overview

- Multiple processes for updating applications with Student Information System and Data Warehouse data
- Provisioning resources to students
- Student Active Directory account creation and changes
- Automated creation of “course shells” for instructors using Canvas
What is “Technical Debt”? 

- Created when we take an easier route via a quick fix for a problem or adding new functionality - instead of addressing larger issues in our code or design.

- Sometimes this is necessary for proof-of-concepts, or the only choice due to higher priorities... but repeating this over time builds the cumulative debt.

- At some point the debt must be repaid, whether due to unmanageable code, staff turnover, security issues such as end-of-life operating systems, etc.
Technical Debt: ATS Student and Course Setup Processes

- Technical debt items
  - Multiple end-of-life servers
  - Hundreds of programs written in older programming languages
  - End-of-life relational database
  - Product latency issues
  - Hard to change code

- How did we get here?
  - Loss of knowledge due to long-term staff retirement
  - Delays with new Student Information and Identity & Access Management systems
  - Organizational resource changes
  - Exciting to start new pilots
Strategies
Getting Buy-In and Where We’re Going

- Get buy-in to address technical debt
  - Inventory debt; decide on an initial plan
  - Engage with decision makers and business stakeholders
  - ... and your own teams

- Where IT Services is headed
  - Integration Platform as a Service (iPaaS)
  - Enterprise Solutions Life Cycle: standardizing development
  - Integrating vendor and open source systems
## Inventory

- Gather processes
- Diagram and document
- Prioritize

<table>
<thead>
<tr>
<th>Description</th>
<th>OS</th>
<th>Location (AWS, On Premises, ..)</th>
<th>Patch Managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Server / VM / NAS / etc.</td>
<td>CentOS 6.10</td>
<td>ITS-VC</td>
<td>Y</td>
</tr>
<tr>
<td>Student Account Lookup Tool: helpdesk front-end for &quot;stuacct&quot;, maildb, etc. similar to ITTools</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Middleware &amp; web server for student resource provision/access</td>
<td>CentOS 6.10</td>
<td>ITS-VC</td>
<td>Y</td>
</tr>
<tr>
<td>Middleware server for student password reset process</td>
<td>CentOS 6.10</td>
<td>ITS-VC</td>
<td>Y</td>
</tr>
<tr>
<td>Primary student account, resource provisioning server (&quot;stuacct&quot;); runs as a Solaris &quot;Zone&quot; (container/vm) atop a pair of Solaris 11 VM hosts.</td>
<td>Solaris 10</td>
<td>SDSC Row 28</td>
<td>Y</td>
</tr>
<tr>
<td>Hosts lab lookup/reservation system</td>
<td>Ubuntu 18.04</td>
<td>ITS-VC</td>
<td>N</td>
</tr>
<tr>
<td>Stuacct Linux dev server</td>
<td>CentOS 6.10</td>
<td>ITS-VC</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Public facing/UCSD facing/internal</th>
<th>can be retired</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal (or ITS facing)</td>
<td>NO</td>
<td>KEEP (active use) - back end to stay public</td>
</tr>
<tr>
<td>public</td>
<td>YES (per ryan 7/2022 find email by entering pid; linked from public)</td>
<td></td>
</tr>
<tr>
<td>internal</td>
<td>NO</td>
<td>SSO (sdacs:sdacs/secure-ad/)</td>
</tr>
</tbody>
</table>
- Design overall architecture
- Think short, medium, long term
- Establish and align with best practices
Develop plan

- Design overall architecture
- Think short, medium, long term
- Establish and align with best practices
- Design overall architecture
- Think short, medium, long term
- Establish and align with best practices

More about **Grouper**

---

**CSPM Architecture - Key Principles**

- Near-term: stuacct feeds AD; pivot middleware to read from AD to (canvas, flatfile, DSMP, etc)
- Medium-term: grouper feeds AD rather than stuacct; middleware remains unchanged.
- Far-term: pivot middleware to pull from Grouper directly (or via Kafka/etc) rather than AD.

---

Add middleware/ETL stages (iPaaS or local)

Does AD offer group state change event subscription?

Does Grouper offer group state change event subscription?
Execution: Following ITS Best Practices to Create Campus Assets

- Agile approach
  - Flexibility and iteration
  - Collaborating with stakeholders

- ITS Enterprise Solutions Life Cycle (ESLC)
  - Data and process flow diagrams
  - Maintainable code: unit and integration testing
  - Continuous Integration/Deployment pipelines
Execution: Following ITS Best Practices to Create Campus Assets

- ITS Integration Platform as a Service (iPaaS)
  - Adopting industry standards
  - Solution type: containers or lambda functions?
    - Containers:
      - Determine parent docker image
      - Create archetype assets that can be used by others
      - Applications are built using archetypes
Execution: Following Best Practices in IT Services to Create Campus Assets

### Python FastAPI API Archetype | Description

Python FastAPI API Archetype is a Python-based docker container built with FastAPI. This version of the archetype supports a fully configured FastAPI web service application. Features include:

#### General Features
- Designed to host REST APIs using FastAPI framework 0.78+
- Environment variables via Pydantic and python-dotenv
- Built-in customizable Health Check
- Built-in OpenAPI 3 support producing automated Swagger documentation
- Custom REST JSON exception handling automatically produces an errors array.
- RESTful API endpoints
- Database support via SQLAlchemy
- Deployable via CI/CD pipeline to EKS.

#### Future Features
- Generate Swagger via default configuration; override via swagger_ui_parameters argument
- Wire logging via python Requests library
- Asynchronous Gateway Interface (ASGI; async-capable for python web servers and applications) via uvicorn worker
- Support for WSGI applications such as Flask/Django/Dash
- Aligned fully with ITS IPaaS goals and strategies

- Better logging
- Example unit and integration tests individually utilizing pytest
- Align with Datadog monitoring and alerting
- Searching pattern
- Pagination
- HAL

- Aligned with Datadog monitoring and alerting
- Designed for Cloud Kubernetes projects with Docker / Container ready framework including embedded Dockerfile and Docker Compose files
Execution: Following Best Practices in IT Services to Create Campus Assets

Getting Started

Your team will need to have the following to make full use of the archetype all the way to production. Submit a ticket to the BARS team for set-up.

- Bitbucket Project and repository
- Bamboo Project space
- Artifactory Docker repository
- EKS namespace

1. Download the archetype code from TBD Bitbucket repository and unzip it.
2. Initialize a new Git repository using git init.
3. Read the README to understand the project structure and start implementing your API.
4. Commit and push as necessary. Be sure to create your own Bitbucket repository and set your remote.
5. Assuming Python is installed, you can test locally
   a. First install requirements python -m pip install -r requirements.txt
   b. Set-up a .env file in the root folder with the minimum required settings. See app/config/environment.py
   c. Start Uvicorn server: python -m uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
   d. Use Postman to make API calls to 127.0.0.1:8888/{basePath}/{routePath}
6. Or deploy to EKS dev via a CI/CD pipeline. See the CI/CD pipeline section below.

CI/CD Pipeline

Follow the sample archetype pipeline at TBD as a guide. Configure your build project, build plan, and deployment project variables appropriately for your team. Do not use the archetype values.

The pipeline expects a Helm 'values.yaml' file to configure EKS, including passing environment variables and secrets. See the included values.yaml in the archetype. Values can either be hard coded (non-secrets, non-environment dependent e.g. application description), or placeholders to be filled in by Bamboo at deploy time (e.g. dev vs production database credentials). See the sample variable substitution script in the deployment project. DIS is creating a standard Helm chart for developers to use, so expect more documentation on that front when it's released.

Please submit bugs to, or request additional information from the Python Community of Practice channel in the DevOps Roundtable MS Team:
https://teams.microsoft.com/l/channel/19%3a29e6a9678074438096385f1cb34058a7%40thread.tacv2/Python%2520CoP?groupId=c49c10f2-43f0-4840-99e67-80d95d5db4d4tenantid=8a188873-4f8c-4e76-8182-ca4760bd6d60
Wrap up: General Recommendations and Resources

- Collaborate with other teams; follow best practices/standards where possible
- Change management - operational and strategic
- Book: “Kill it With Fire - Manage Aging Computer Systems and Future Proof Modern Ones”
Contact Us

Ryan Rose - rmrose@ucsd.edu
Paul Jamason - pjamason@ucsd.edu

Questions?
TECH Talks

Catch your next TECHTalk on the third Wednesday of each month at 11:30 a.m.

Questions or comments?

tech-feedback@ucsd.edu