**Introduction (½ hour)**

Introductions (What does success to you look like?)

* Why are we here? Describe what an inception is:
  + Meant to be a discovery exercise & to get everyone on the same page.
  + Things can change (a lot) after the inception. Watch for scope creep/change.
  + Not a silver bullet.
  + Mention **Domain Language Dictionary** (on wall).
  + Quick philosophy of app replatforming and modernization.
* Timeline of the replatforming. Discuss breaking up the engagement into two/three parts with breaks in between. Prime the possibility of a pivotal labs location for the last week
* Roles (who's who on the team): *Customer Team*: The team that operates and runs Cloud Foundry, Executives and PCF supporters, Developers who will pair with Pivots, Architects, Test team, Business domain experts. *Pivotal Team*: Inception-Master of ceremonies, Anchor and 3 Pivot developers/architects.

**Culture Introduction (½ hour)**

The way Pivotal does delivery in a DOJO: TDD, Pairing, MVPs, Dedicated-time, Roles of Pivots, IPM, retrospective, Stakeholder communication.

**Goals (½ hour)**

Group discussion - list on paper; move paper to wall

Who are customers & stakeholders? Who shouldn't be your customers and aren't your stakeholders? What are your products?  What shouldn't be?  What is success? Disagreement and vibrant discussion is OK.

* Goals - covers both business & technical (e.g.: Migrate apps of a certain type, Generate cookbooks)
* Non-goals - highlights process issues (e.g.: seek perfection, Dockerize apps)

**Portfolio Analysis/Affinity Group Mapping (½ hour)**

Derive an understanding of the whole portfolio based on like technical stacks such as JavaEE Web Profile and Full Profile apps, Spring apps, non-java apps, Mainframe, dotnet, RoR, NodeJS, Javascript. Dissect the entire portfolio to come up with a list of 10/15 apps to snap analyze. The intent here is to pick a variety of apps from each category of the portfolio. This can also occur during the scoping of the engagement and is repeated here on a deeper technical basis if we are choosing from 20-40 apps

**SNAP Analysis of Apps (½ hour per app)**

**Topics to ask about to help with characteristics of an Application:**

|  |  |
| --- | --- |
| * Java, JavaEE, J2EE & Spring Framework versions & APIs | * External Integrations - Databases, Caching, Messaging Queues |
| * In-house frameworks | * Logging |
| * Configuration | * SLAs |
| * Packaging and Build (ear/war/jar) | * Is CI in place? |
| * Use of Distributed 2PC XA Transactions | * Persistent File System Access like NFS or SMB/CIFS |
| * People location | * non-HTTP inbound networking protocols like RMI-IIOP, etc. |
| * Total LOC | * App Server ? and appserver specific code |
| * Security Requirements | * Batch/ETL |
| * Front-end tech | * Runs on PCF |
| * Statefulness - session state | * Data Access - JDBC, ORM, JMS |
| * Startup/shutdown times | * 3rd party frameworks/libraries- 32/64 bit |

References - [bit.ly/app-replat](http://bit.ly/app-replat) & [bit.ly/migrate-jee](http://bit.ly/migrate-jee)

**Outcome:** (0.Characteristics, 1.Success Metrics, 2.Risks, 3.User Stories, 4.Score aka Summary))

Outcome for each app is a list of stories to get the app to degree of cloud nativeness desired by customer. Each App maps to multiple MVPs. First MVP is always getting the app running on PCF. Thereafter there can be multiple MVPs gradually moving the app on the modernization scale. The scorecard of each app is used to map all the apps on a quadrant of Business Value/Technical Risk. Apps of low-tech risk and high business value are chosen first.

**Planning/Delivery**

Create a Kanban board of [Candidate apps | Started | Finished] on a week-by-week basis for the 10 weeks. This is mainly to map people to apps and determine scheduling.

Always start apps of Low technical complexity first to demonstrate quick wins.

Select a mix of apps for replatforming and modernization.

Remember to set modest expectations and most important of all keep the deliverables small.

**Risks**

Individuals write their risks on cards. Stick cards on paper, grouping similar risks together. Move the paper to a wall for easy visibility. Figure out mitigations for risks. Get executive buy-in. This is especially critical for platform blockers (e.g.: firewalls or provisioning of services from the CF marketplace).

**Retrospective**

Were people's goals met? What went well/what could have been better. Actionable Items.

Email a summary deck of the retro after the inception.