Self-Service Microservice Platform Project Overview

Justin Smith

August 27, 2021

Summary The integration team at [redacted] is tasked with creating a self-service platform for other internal development teams to use to develop, host, and manage microservices. Now the team needs to take the steps necessary to kickoff the project and create a product roadmap. This document can be considered a "roadmap to the roadmap".

Establish a Charter

“Plans are worthless. Planning is essential.”
— Dwight D. Eisenhower

The first step in any project or program is to establish a formal charter for the development of a minimum viable product (MVP). The charter includes the following items[1]:

[1] It’s a good idea to give the project codename or working name, such as [redacted], [redacted], [redacted], [redacted], [redacted], [redacted], [redacted], [redacted], [redacted], etc. This anchors the project and makes internal marketing easier. It can also be helpful to establish a name for the project team (i.e. *skunkworks*), which helps build camaraderie and esprit de corps.

- Project description
- Purpose and objectives (expected outcomes) of the project
- Project vision: the capabilities we imagine will be available when the project is "done", perhaps several years in the future
- Feature requirements of the MVP
- Make or buy justification for the project
- How long we expect personas to be satisfied with the MVP
- Expected major release cycle (quarterly, annually, etc.) after MVP
- Under what circumstances the project becomes abandonware
- Known high-level, major categories of risks
- Schedule of events with the start and end dates
- Key events (milestones)
- Budget or summary of how much the project will cost
- Criteria for judging the project to be a success or failure a year from now? (Licensing income, daily average users, etc.)
- Requirement for approval, including what to approve, who approves what, and how to get approvals
- Key players or stakeholders and which parts of the project they will touch
- An introduction of the project manager, project sponsor, and their authority

- The current team members and planned hires, if any

Most of the charter content is a guess at this point. The charter doesn't need to be a very profound document, but it's existence is required to ensure involved parties have reached consensus (or "aligned expectations") on the most basic items before doing any work. [2] It's ok to revise the charter as needed, but each revision should be signed by the sponsor.

[2] There is a general movement, especially in tech, towards doing absolutely everything on a computer. But this is a mistake. The key project documents, starting with the charter, should be printed on paper, signed, and kept in a binder. The impact of physical permanence and tangible cost is almost always underestimated and underrated.

Key Event: Define Requirements for MVP

We need to understand the nature of our *customer* and what problem or pain point we're solving for them. Often, the "customer" is actually several different people, who may or may not be end users of the product. So this term is quite ambiguous, and we should take care to be more specific around the who, the why, and the when.

It's helpful to first compose a list of questions, which are then answered by the project team. For example:

- What basic features should the MVP should have at launch?
- Who are the initial target end users of the MVP? Are they employees or contractors (if security and access are different)? Where are they seated? What kind of problems are they currently trying to solve? Will they collaborate on development with us in some way?
- How many users will we have at launch? How many will we have six months after deployment?
- What new features do we think will be added over the coming quarters and years?
- How will the product features be documented? How will changes to the product be communicated to different stakeholders?
- How will the product be marketed?
- What kind of license will be applied to the software?

The answers should be recorded in a table or spreadsheet and appended to the project charter.

Key Event: Create Personas for MVP

> "The single biggest problem in communication is the illusion that it has taken place." — George Bernard Shaw

From these answers, we can create a number of individual *personas* for both real and imaginary customers. Whenever a question comes up about what the "customer" or "end user" would want during the course of development, we can simply refer back to our personas for guidance.[3] The project team will reach consensus on each persona before proceeding, and the personas appended to the charter.

[3] It's a good idea to have at least one persona, but not more than five. The number of personas, and their detail level, should put a natural soft constraint on the scope of the project.

Key Event: Define Tooling, Architecture, & Documentation

After our MVP requirements have been established and our personas have been created[4], the technical team needs to make a number of lower level decisions. Which programming language(s) they plan to use (golang, java, c++, lisp), how they will collaborate (github, gitlab), how issues will be reported and tracked during development and after launch, versioning format (vX.X.X vs YYYY-MM-DD), deployment schedule (e.g. never release on a Friday), who will build and maintain the pipeline, how is documentation performed and made available to customers (manual internal wiki updates, third party tools that parse comments in the code), and so on.[5]

[4] Meaning the requirements have been frozen. We of course understand that requirements may change...but that's what project management is for.

Some obvious choices for what could go into the tooling stack[6] include:

[5] Sometimes you start using a framework or tool only to find out later that it's not going to work. The person doing the work should make the call.

[6] Respecting the UNIX philosophy, see http://www.catb.org/esr/writings/taoup/html/ch01s06.html

- Gitlab for version control, kanban, and issue tracking.
- Docker / Docker Hub. Fast deployments and testing, and portable. Docker is probably stable enough for an MVP. Potentially some learning cost involved.
- OpenBSD. Stable and secure for production deployments. Potentially some learning cost involved, and likely cannot use Docker (natively anyway). Depending on productivity level of DevOps, Debian or Ubuntu may also be an option.
- [redacted] [7]
- [redacted] Depending on the team's ambitions and quality requirements, consider taking a functional programming style when possible. There is no real need to be religious when it comes to language choice(s), as long as the product requirements are met.

[7] It could be worth investigating AWS and Google Cloud for the generous free credits they tend to offer new customers. This would also help prove the portability of the product.

There are likely many more tools here that could be worth investigating, such as IBM's Istio. But these are much lower level decisions that should be taken only after extensive consultation with the technical team. Preference should be given to FOSS, and re-inventing the wheel should be avoided.

Whiteboarding sessions are likely necessary to create block diagrams, interaction diagrams, sketches of the MVP's UI, etc. which begin to formally describe the architecture. The draft platform architecture documents, including any diagrams and other low level specifications (e.g. object oriented or functional programming styles, thread safety, special security concerns, etc.), are appended to the charter.

Key Event: Marketing Plan

How will the product be advertised and promoted? Will we create a marketing webpage? Will we create an explainer video? Send out an email blast to prospective users? Call customers individually? Who will be responsible for doing these things? A draft marketing plan should be established at this point. This is a one page or less agreement explaining how the team plans to help people discover and learn about the product. When completed, this is appended to the charter. Depending on the scope, marketing may be a sub-project that will have its own critical path.

Key Event: Create the MVP Product Roadmap as a Gantt Chart

At this point we can create Gantt/PERT charts, assign resources and guess timings, and establish milestones for development. This will include testing and certifications. This roadmap will be tracked and updated by project leadership, while the developers and engineers follow their own methodologies for delivering the results they are accountable for.[8] As the project runs, weekly team meetings for updates should be sufficient for tracking progress and issues.

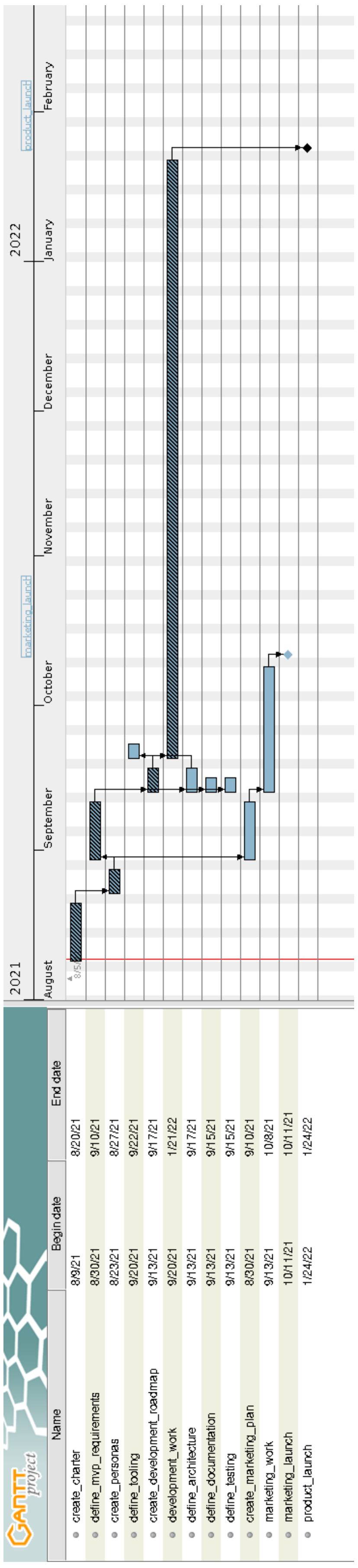
[8] See example Gantt chart.

When the Gantt chart is completed and approved by project leadership, the first version of the chart is appended to the charter and the team can begin to work on the project. Each subsequent revision to the chart is appended to the project charter, until the MVP is delivered.

Key Event: Development Work & Product Launch

If all goes as planned, the MVP should be delivered on time and on budget. The reaction of the intended users should be observed and documented and appended to the charter. The team should also complete a retrospective, and a final report appended to the charter. The project binder containing all of the project documentation should be archived.

New features should be planned and implemented by repeating this entire process, starting with a new charter, as needed.



Name	Begin date	End date
create_charter	8/9/21	8/20/21
define_mvp_requirements	8/30/21	9/10/21
create_personas	8/23/21	8/27/21
define_tooling	9/20/21	9/22/21
create_development_roadmap	9/13/21	9/17/21
development_work	9/20/21	1/21/22
define_architecture	9/13/21	9/17/21
define_documentation	9/13/21	9/15/21
define_testing	9/13/21	9/15/21
create_marketing_plan	8/30/21	9/10/21
marketing_work	9/13/21	10/8/21
marketing_launch	10/11/21	10/11/21
product_launch	1/24/22	1/24/22

Figure 1: Example Gantt Chart with critical path highlighted.