

Onboarding Documentation

Our Mission

Here at JamSession, we aim to support content creators by connecting fans of the genre and similar artists to them. By doing so, we can expand the listener's musical libraries to have a near limitless selection of content to pick from. As we get to know the listeners better, we are able to work with various ad platforms and match them with their appropriate audience.

Priority #1 - Do What's Best For the Customer

Every change we consider for the JamSession platform has to first consider if it will bring any value to the listening audiences, content creators, or our advertising partners. Changes that benefit the stakeholders should be considered a core feature that gets prioritized. If a proposed change will impact any of these parties negatively, not a change worth doing. It's crucial that we work with our customers to ensure the tools we provide them will be successful in their own use. If they are successful, we are successful.

Priority #2 - Go Wide

With a developing marketplace full of competition, it's important that the JamSession platform is flexible and able to adapt to changes in the space. The team can accomplish this by taking a breadth-first approach to implementing any features to the JamSession platform. Overcommitment to a single feature will result in a failed feature.

Priority #3 - Make it Simple, then Make it Good

As we develop new technologies for JamSession, we want to make sure the features we make are actually going to be used. To accomplish this, features that are being worked on should be built with modularity in mind. Within these modules, they should be able to stand on their own at the end of an iteration cycle for review to determine if the feature is bringing value to a customer. If it isn't bringing value, we can either modify it so it can be of value, or remove the module from the platform without harming the rest of JamSession.

Our Day to Day Operations

Agile Hybrid Methodology

Recognizing the value of iterative and modular development, we wanted our day to day workflow to reflect this. However, instead of landing on one type of process framework like Scrum, XP, or Evo, we recognized that there are valuable parts to each framework. With that in mind, we took those parts and embraced an Agile Hybrid approach to the SDLC that is modular in design.

Iteration Lengths

Iteration lengths are generally 3 weeks. We found that the one to two week cycle from Evo was far too short, and the maximum month time-length of a long Scrum sprint was far too much to plan for, while having features be flexible. Tasks need to be able to be completed within the iteration length. If a task is too big to fit into an iteration, it needs to be broken down further so it can fit.

The Bridge Builder

The “Bridge Builder” is a position designated at the start of a project. The Bridge Builder is responsible for facilitating adherence to processes during the SDLC, assists the team when they are stuck, communicates changes to and from the stakeholders, and works with management to ensure the support of the project.

Prioritizing Work

During the planning of an iteration, the team should get together and review what is on the table for the next iteration. Prioritizing work is done based upon two categories, **Technicality and Impact**.

Examples of Technicality

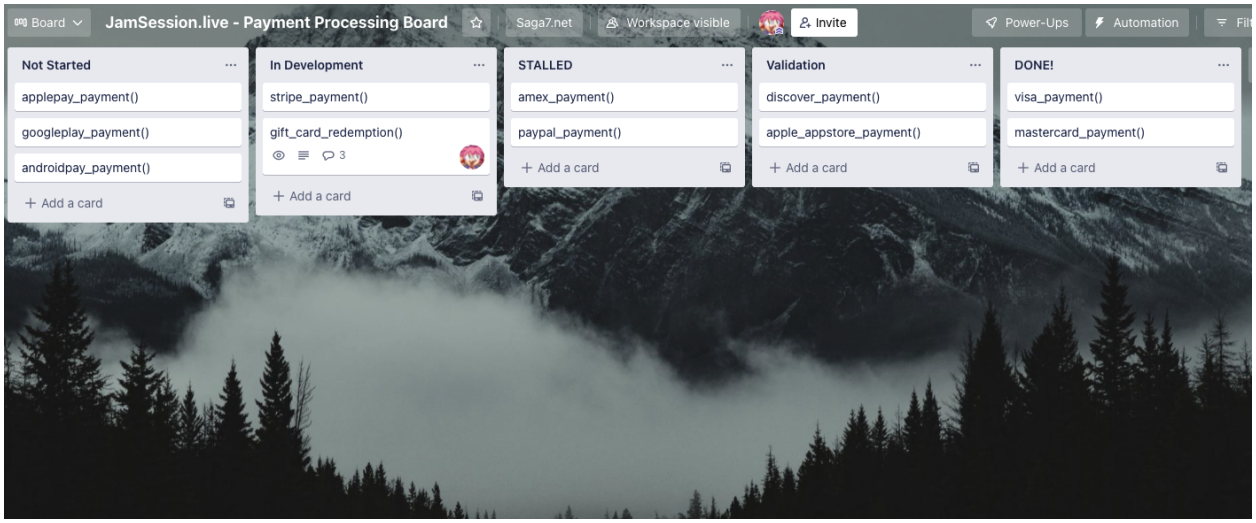
- Do we have the resources to complete it in an iteration?
- Is the roadmap of this work easily identifiable?

Examples of Impact

- How does this help our stakeholders?
- How does this harm our stakeholders if not completed?

Kanban Board

Each iteration has a Kanban Board that will have all of the work that is needed to be done in that iteration.



Kanban Board for the Payment Processing Feature

The tasks are individual cards that should have the following information:

1. Task Name
2. Function Summary - A brief summary of what the the goal of the task is
3. Complexity of the Task - Rated on a Low/Medium/High Scale
4. Dependencies - Anything outside of the feature that is required for the feature to function. This can be things like servers, APIs, databases, etc.
5. Holdover Status - If the task is from a previous iteration that wasn't completed in time, it is considered a holdover item. These are crucial to get done as soon as possible.

Meetings

Scrums

Meetings during an iteration usually come in the form similar to Scrum meets. These occur daily, are headed by the bridge builder, and the goal of each scrum meet is to learn the following:

- What was done yesterday?
- What is being done today?
- Are there any blocks?

These meets are a great way of synchronizing the team, and ensuring any issues that are in the way are out in the open to get addressed.

Reviews

Reviews are a long form meeting that focuses on the impact of the topic. Sometimes they are related to software quality assurance, other times its data collection. Processes will infer whether or not reviews are part of the process.

Demos

Demos are a special type of meeting that occurs with a stakeholder. Usually demos occur towards the end of an iteration. Team members that should be present are:

- Primary developer working on the proposed feature
- Manager of the team (not necessarily the Bridge Builder).
- Any management that would have a stake in this feature

Demos are a great way of getting feedback on how a feature is turning out. Your goal of a demo should be to learn the following:

- What did the stakeholder love?
- What did the stakeholder hate?
- Did the stakeholder explicitly state any desired features or changes, and what are they?

These three facts will be key in shaping the next iteration, and how many iterations more are needed until the project is fully realized.

Our Processes

Software Quality Assurance

At JamSession, quality is baked into everything we do. Whether it's how a button looks, up to our backend design, software quality assurance checks are in place at every step of the way is important to our scalability and reliability for our services. Below is our six-point plan to ensure SQA goals are being met.



JamSession's Software Quality Assurance Roadmap

1. Have a Plan

Before even starting quality assurance checks, understanding the environment is just as important as the quality itself. Things to consider in this phase are:

- Why are we evaluating this?
- What is being produced?
- What is the current quality being outputted?

2. Plan the checkpoints

During the execution of a process, ensure that there are checkpoints or stopping points to evaluate if the process is adhering to the standards and goals of the company. Things to consider when placing checkpoints:

- Is this a natural point to stop and evaluate?
- How often should checkpoints take place
- Will placing a checkpoint halt activities?

3. Work and Estimate
While working through a process between checkpoints, data collection should be done to support if quality work is being outputted and the estimated time to completion. Data should be collected and categorized in either "**Project**" or "**Process**". Some data points to consider are:

Project

- Burndown Rate
- Backlog
- Defect rate per iteration

Process

- Time to release
- Developer SLOC written
- Defects found after being released

4. Review
During the review phase, staff working on the project are brought together and discuss the requirements of the project and any prototyping, and where it sits in term of its quality level. These reviews are a way to change bad habits and unify standards resulting in less reworks and errors in future iterations.

If needed, conduct additional private reviews for staff that need it.

5. Test
In our Quality Assurance approach, testing needs to be done in multiple ways to ensure robust, quality products are being delivered to our clients. Some methods of testing are:

- Unit Testing
- Insider-First Beta Testing
- Quality Assurance Team Testing

6. Report
This final phase consists of keeping records of everything you've done so far up to this point. Records should be indexed and searchable in a digital location that is accessible to any user that would need access to this. Reports are eye openers to the following:

- Do we need to change tools?
- Do we need to change the process?

Being Secure through Risk Management

In every process at JamSession, risk must be assessed to determine the security of the platform. Risk is interpreted in a matrix of how likely an event occurs, and the impact it makes on the JamSession platform. The matrix result will determine the security of the system.

Likelihood is defined as the possibility of some event happening. This is evaluated in four increasing levels:

- Rare
- Possible
- Likely
- Assured

Impact is such an event occurred, what would the harm be to the JamSession platform. This is evaluated in four increasing levels:

- Minor
- Moderate
- Severe
- Critical

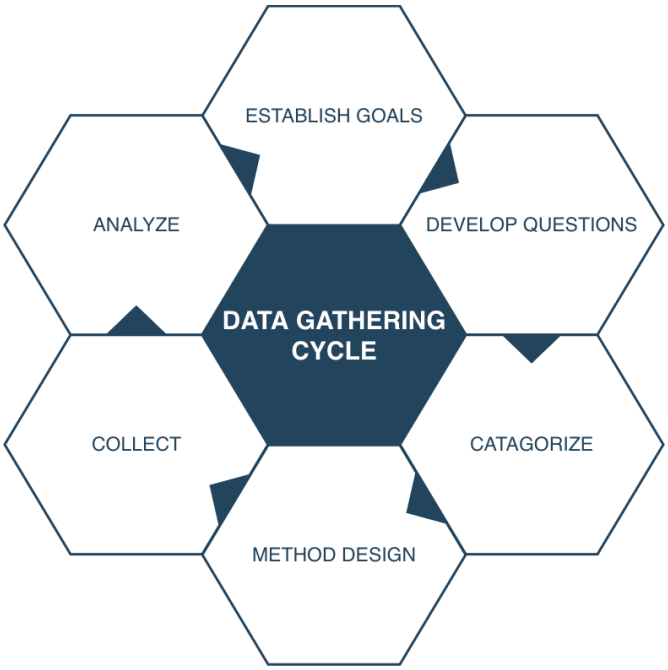
Likelihood	Minor	Moderate	Severe	Critical
Assured	Medium	High	Urgent	Urgent
Likely	Low	Medium	High	Urgent
Possible	Low	Medium	Medium	High
Rare	Low	Low	Low	Medium
	Impact			

JamSession Risk Evaluation Chart

The matrix gives us a better way to plan for the future of the platform. “Act of God” type events like a volcano forming and simultaneously erupting at our datacenter would certainly be devastating for JamSession, however, the likelihood of an event such as that is so low, we don’t need to plan around it. Inversely, small issues like “player takes .2 extra seconds to load when resuming from an idle state”, is an issue that is likely to happen, but it’s such a minor issue that resources shouldn’t be allocated to it until riskier issues are addressed.

Data Gathering

At JamSession, we are always collecting data as part of our feedback driven approach to designing features that give value to our customers. Amassing data is important, but getting the *right* data is even more important. To do this, we follow a six-step cycle for data gathering.



JamSession's Data Gathering Life Cycle

1. **Establish Goals**
What do we want from our data collection efforts? Is it to improve the discovery of artists? Is it to better define user demographics for our advertisers? Whatever it may be, a goal must be established before continuing the cycle.

2. **Develop Questions**
With the goal in mind, we begin to formulate questions to support reaching the goal. "How long are users listening to artist X" or "what is the clickthrough with advertiser X with demographic Y" are some example questions to consider.

3. **Categorize**
Now that we have some questions in mind, it's important to give these questions a form of classification. Are they quantitative? Qualitative?

4. **Method Design**
Now that we have our questions sorted, we need to design a method that can have these questions asked. Some design approaches to be used could be:

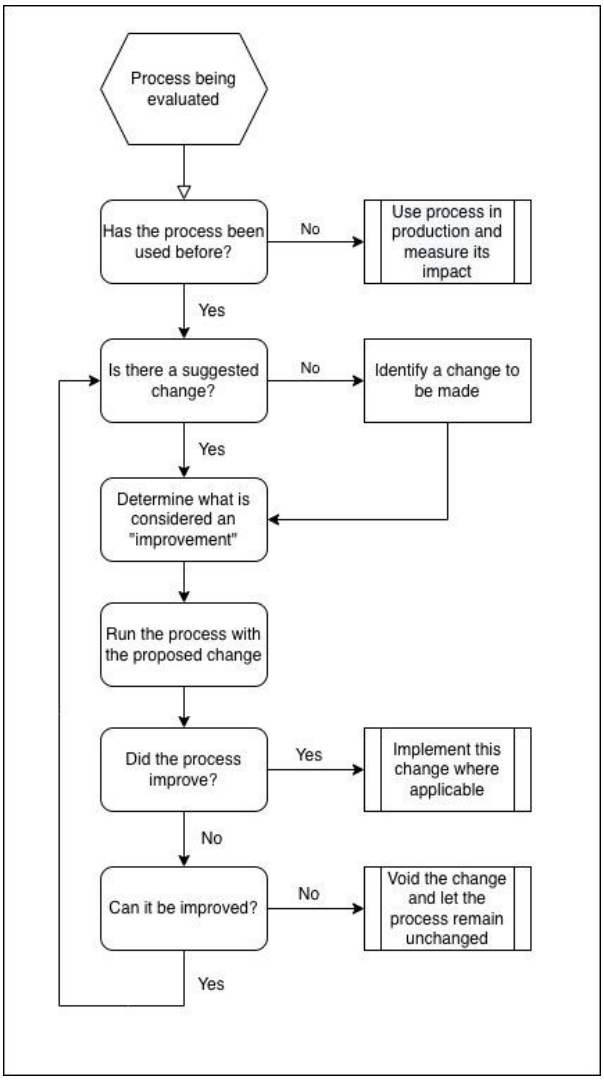
- User surveys
- Likes/Dislikes count
- Listener Retention

5. **Collect**
With our collection methods designed and ready, go out there and use them to collect the data!

6. **Analyze**
Once the data collection has been completed, data is to be sorted into those categories we made earlier in the process, and is then analyzed. The data collected might be enough to meet our goal, but if the results are inconclusive, iterate through the cycle again with a different approach.

Process Improvement

Very rarely are things perfect when created the first time. Much like our development teams, we have to iterate without our processes and improve them when possible. Below is a diagram of how anyone can improve a process at JamSession.



JamSession's Process Improvement Flowchart

In each phase of process improvement, it's always important to ask questions.

The first question to ask is "Has this process been used before".

- If the process hasn't been used before, we cannot meaningfully measure its impact. Changing it before understanding how it works in practice could harm its impact.

After we've determined the process has been used, we should ask "Is there a suggested change?"

- If no change has been suggested, a change should be identified by picking parts of the process to change, and how to do so.

Knowing what change is going to be made, we should ask "What is considered an improvement". By doing so, we can measure the impact of the change and determine
if this process evaluation was a success!

Once we understand what will be a successful change, it's time to run the process with the proposed change. This trial will dictate if the proposed change is beneficial.

- If the process was improved, then congrats! The process modification was a success and any documentation related to said process should be updated to reflect the change, and the teams are notified of the change.

Sometimes, changes to processes may not be considered a success. It’s important to reflect if the process we are working with can be improved even though this change did not work out. If the team can see an improvement, repeat this improvement process.

- If no further improvements can be realized at the time of review, the process should not be changed, and re-evaluated at another time.

Operational Processes

SRE is our Bread and Butter

Here at JamSession, our service model requires a robust operations center that can adapt and scale as our operations grow. With an advanced tech stack, it encourages that our operations team has a deep understanding of software engineering. As a result., our solution to running the operations of JamSession falls upon the JamSession **Site Reliability Engineering** (SRE) team.

Site Reliability Engineering is what you get when we take Software Engineers and put them into a position of operations. They are the bridge between a traditional IT operations team, and the software development team. The primary goals of the SRE team is:

- **Monitor** the reliability of systems and improve as needed
- **Predict** failure points in the environment and resolve these issues
- **Automate** operational tasks with approved software
- **Optimize** the environment with speed, performance, and reliability in mind

With these goals in mind, the SRE team will execute on these tasks with adherence to the service-level agreements set in place between JamSession and its customers.

SRE teams are responsible for how code is deployed, approved, and pushed to production. In addition to code management, SRE teams maintain the availability, latency, change management, emergency responses, and capacity planning of the environment. Toolchains for both of these responsibilities are in the “Operational Toolchain” section of this document.

SRE team members are encouraged to learn more about how the industry is shaping and evolving with changes in demand. If a process isn’t in spec with how JamSession is operating, the SRE team can submit a formal process improvement request outlined in the previous section to determine if the change is meaningful, fits within the goal of the SRE, and can be executed while adhering to the SLA.

Service-Level Agreements (SLA)

About

The Service-Level Agreement (SLA) described here is JamSession's promise to our end-users that our service will meet. The SLA will identify the services required and the expected level of service operation within a given period.

Type of Service to Be Provided

JamSession is a content delivery platform targeted at delivering music via streaming to its end-users, and maintaining a community of music listeners. This content delivery is segmented into the following categories:

- Music Player
- Playlist
- Social Media Sharing
- Album Art
- Advertising Delivery
- Customer accessible API

Monitoring Process

The Monitoring Process is done through the SRE team. Monitoring is influenced by the Service-Level objectives set by JamSession. Auvik is used to gather metrics on our infrastructure, and this data is displayed with Tableau dashboards. Dashboards are hosted on the status.jamsession.net site and are available for public viewing.

Response and Issue Resolution Time-Frame

Incident response and Resolution Time-Frame is a priority for JamSession. Incidents in the service are rated on a five point priority (P) scale.

- P1 - The highest severity level. This type of incident indicates all services are offline and inaccessible to outside users.
- P2 - This type of incident indicates that more than one service is offline
- P3 - This type of incident indicates that one service is offline **OR** services are heavily degraded

- P4- This type of incident indicates there is a degradation in the service that is noticeable by an outside user
- P5 - The lowest severity level. This type of incident indicates there is an incident that is isolated to a small subset of users and isn't noticeable by the affected users. These incidents are considered "inconveniences".

Below is an outline of our common response times and resolution time frames

Action	Response Time
Automated monitor triggers alerts to an anomaly	5 minutes after detection
SRE Team reacts to an alert	Within 10 minutes
SRE Team initial diagnosis and designates sev level	Within 60 minutes
80% operational recovery from a P4 incident	Within 4 hours
80% operational recovery from P1/P2/P3 incident	Within 6 hours
95% operational recovery from P4 incident	Within 6 hours
95% operational recovery from P1/P2/P3 incident	Within 12 hours
100% operational recovery from any incident	Within 24 hours

Repercussions for Service Provider for Not Meeting Commitment

In the event JamSession is unable to meet the performance level within a given period outlined in the SLO, compensation will be provided to customers impacted. Compensation is offered depending on the customer impacted. Compensation methods are outlined below:

- End-User: 1 Month Free of JamSession+
- Content Creator: Algorithm promotion to increase likelihood of play in discovery feeds
- JamSession API Users: Credit 1 month of API user service to the user
- Advertisers: Credit account equal to cost incurred for ads that were served outside of performance level indicator

Service Level Objectives (SLO)

The Service-Level Objectives (SLO) is the quantifiable requirements for JamSession to meet to ensure we are within the agreements described in the Service-Level Agreement. How to measure the Indicators is defined in the SLI section.

Indicator	SLO Requirement	Measurement Period
Availability	Service will be available for 99.91% of the period	Over 365 days
Latency	80% of external requests are served within 200ms 99% of external requests are served within 100ms	Over 30 days
Quality	99.91% of all valid requests are processed without degradation	Over 180 days
Throughput	90% of data delivered at 96kbps 99% of data delivered at 320kbps	Over 90 days
Response Time	80% of data received is processed within 1000ms 99% of data received is processed within 2500ms	Over 30 days

Service-Level Indicator (SLI)

The Service-Level Indicator (SLI) is the direct measurement of various metrics to evaluate if we are running within the Service-Level Objects (SLO) within a given period. The categories we use to evaluate SLO status are: **Availability, Latency, Quality, Throughput, and Response Time.**

Availability

Availability is defined as: “A segment of valid requests that have been served successfully”.

Availability is arguably the most important indicator to our service. If the service is unavailable, we are unable to measure our other indicators for end-user metrics. We keep count of HTTP response status codes delivered to users. 200-level codes are indications of successful responses, while 500-level responses count against our SLI. 400-level responses are not included in the availability calculation.

Availability is calculated as such:

$$Availability = \frac{Sum\ of\ HTTP\ 200{-}level\ Responses\ Delivered}{Sum\ of\ HTTP\ 200{-}level\ Responses\ Delivered + Sum\ of\ HTTP\ 500{-}level\ Responses\ Delivered}$$

Latency

Latency is defined as: “A segment of processed valid requests served above a threshold”.

Latency is a measure to determine if a service is fast enough, and can be measured numerically. However, a more critical point of latency if it is **user-detectable**. If we are meeting an SLO numerically, but the user detects that a service is slow, we are effectively failing to meet our SLO.

Latency should be measured in two (2) separate instances. These are:

- Initiated - The time it takes to serve a result that has been requested by a user.
    - Example: User selects a song to play.
- Background - The time it takes to complete an action that is not noticeable by the user.
    - Example - Preloading thumbnails outside of viewable space, but within the “buffer zone”.

Latency is calculated, in ms, as such:

$$Propagation\,Time \;=\; Frame\,Serialization \;+\; Link\,Media\,Delay \;+\; Queueing\,Delay$$

Quality

Quality is defined as: “A segment of valid requests served without the service being degraded”.

Quality is not something that cannot be measured directly by any machine, but a result of how a user *feels* about the service. If a large enough group of users feels that the service is not living up to the operating standards, the service is considered degraded. To determine degradation, we establish the **User Feedback Threshold (UFB).**

- User Feedback Threshold - The number of users voicing concerns about the quality of the service through electronic means over a sampled period of time.
    - Example: Number of complaints over social media and keywords “JamSession” + “down/dead/not works/not working” over the past 10 minutes.

If the value produced from the UFB exceeds 50 complaints per minute, service is considered degraded and an investigation is initiated. This time is measured in minutes. Upon restoration of service, the service is no longer degraded.

Quality is calculated as such:

$$Quality = \frac{Requests\ Served\ without\ Degradation}{Period}$$

Throughput

Throughput is defined as: “A segment of time where the data processing rate is faster than a threshold”.

Throughput is the speed in which we are delivering content from the server-side host to the client-side destination.

Throughput is calculated as such:

$$Throughput = \frac{Data\ Rate}{Time\ (in\ seconds)}$$

Response Time

Response Time is defined as: “The time taken to process a valid request”.

Response time is the result of how fast we can process data computationally. It is a measure to gauge our computational power in tandem with efficiency built into our algorithms to ensure we can meet the demand of the users. Response time requires two important data pieces, **Time Received and Time Completed.**

- Time Received - The timestamp indicating when a computational system has received the request from the network.
- Time Completed - the timestamp indicating when a computational system has processed the request and is sending it to the destination.

Response Time is calculated as such:

$$Response\ Time\ = Time\ Completed\ -\ Time\ Received$$

Operational Toolchain

Our tech stack for all of JamSession can be categorized into 5 distinct categories:

1. Collaboration
2. Build Generation
3. Automated Testing
4. Automated Deployment
5. Execution



Employees working within the JamSession organization are authorized to use these tools for production purposes. Tooling not listed must be approved by management. The SRE team is responsible for maintaining uptime on these toolchains.

Collaboration Tools

Collaboration tools are tools that our employees work with in the day to day operations. They assist our teams in being able to share ideas across the company and are the foundation to our operations. Without collaboration, we can't build JamSession to be the service our users need.

Approved Tools

Lifecycle Management - Trello
Internal Chat - Slack
Internal Documentation - JamSession.net Internal Wiki
Knowledge Base and Self-Help - Read the Docs

Build Generation

Build Generation Tools are used by our developers and SRE team to ensure the JamSession service is constantly growing, improving, and most importantly, meeting our customers needs. These tools vary from tools to assist in the software development process, to database management tools.

Approved Tools

Source Control Management - Github
Continuous Integration - Jenkins
Automated Building - Gradle
Database Management - DBDeploy

Automated Testing

Automated testing is crucial in our workflow. Automated testing frees up our teams to continue to build a bigger and better service for our users. Testing is done on a daily schedule to determine if commits are bug free, meet security standards, adhere to our standards, and if they pass can be pushed to the automated deployment

Approved Tools

Web Testing - Selenium
Application Testing - Appium (Selenium for Mobile Platforms)
Load Testing - Terraform.io
Network Testing - Obkio

Automated Deployment

Automated deployment is the natural progression from our automated testing processes. Automated deployment allows for the JamSession service to be up-to-date and maintain relevance in a highly competitive industry.

Approved Tools

Continuous Deployment - Jenkins
Configuration Management and Provisioning - Terraform
Artifact Management - JFrog

Execution

Execution tools are the tools that the SRE primarily interact with. They are mostly infrastructure tools, and tools to support infrastructure. These tools are crucial to the day to day operations of JamSession, as these tools give us indication how well the service is being served to our end users.

Approved Tools

Cloud Platform - Google Cloud
Orchestration and Scheduling - Kubernetes (K8s)
Business Insights - Tableau
Monitoring and Logging - Auvik