Overview of Goals

You should assess Jira's performance and stability by looking at three areas: time to incident response and resolution, general application trust by the business, and a reliable baseline for validating your future state. Think of these as past, present, and future. As these are very different issues we address each one independently in the following four sections:

**Atlas Load Technical Overview** - We discuss what the tool does from a technical perspective and run through a simple usage scenario. The goal here is to lay the foundation for the following three sections.

**Incident Response** - It is critical to have procedures and tools to quickly, effectively, and retroactively analyze an incident. This section covers how to prepare for an outage. Ensuring you have everything you need to quickly and effectively troubleshoot. Much of this is available in disparate places, so we present a cohesive image and extend it with Atlas Load.

**Application Trust** - Trust in the application, changes made to it, and its overall performance and reliability are critical to an organization expanding their usage of Atlassian (and associated ecosystem) tools. To meet this need, Atlas Load can be used in tandem with upgrade testing, demonstrating Data Center as a path forward from existing problems, and regular load testing.

**A Better Baseline** - New components enter your environment on a regular basis. Testing them is going to be difficult and present you with various challenges. Atlas Load can help create a better environmental baseline on top of which you can test new functionality.

- Atlas Load Technical Overview
  - General Architecture
  - Security Considerations
- Incident Response
  - Doing it Wrong
  - Doing it Right
- Application Trust
  - Fixes
  - Regression Avoidance
  - Excessive Risk Mitigation
  - Data Center
- A Better Baseline
- Value
- Freebies - Audit Logging
Atlas Load Technical Overview

Creating high-quality load tests using existing tooling is difficult. Not only due to the complexity of existing tools like Gatling, JMeter, or E3 but also because once you know how to build out the config for these tools, writing tests that match your actual user behavior is very hard. There are no tools that we are aware of that can generate these configurations for you dynamically based on your environment.

The second challenge is that if you step back from the application perspective and look at it from the network level, you have the application security to overcome. A PCAP and replay operation does not work. Validation of security tokens blocks requests, and you do not generate the desired behavior or load on the system.

Atlas Load solves this challenge by taking the network approach and intelligently rewriting requests on the fly to bypass Atlassian’s security mechanisms.

General Architecture

Model of a typical Atlassian environment in a large system:

![Diagram of network architecture](image)

Atlas Load sits on the application server and captures the network traffic, and then allows us to replay the same traffic in a staging environment:

![Diagram of record and replay process](image)

Atlas Load provides a Node.js middleware to make it easy to customize to your specific environment.

Security Considerations

Atlassian tools do not hash passwords before sending them over the wire. To overcome this, Atlas Load overwrites all user passwords with a single hardcoded password before writing to disk. An administrator who reviews these files can never see the users passwords. To utilize this file for replay, we must replace all user passwords in the staging environment database with an appropriately matched hash. If you use an LDAP server for authentication, you need to move users into the internal user directory in your staging environment or create a mirror of the LDAP server where you can overwrite the user’s passwords.
Incident Response

Now that we understand the basics of how Atlas Load works, we can see how it fits into an administrators incident response toolkit and strategy.

Doing it Wrong

After an outage occurs, the team responds by restarting the application to stabilize and then analyzing logs to ascertain what went wrong. In an ideal world, they would immediately identify the problem and fix the problem. However, the reality is typically closer to:

- A requisite log file was not enabled, or the configured logging level is not high enough.
- Troubleshooting tooling was not in place.
- Administrators did not respond to the outage promptly and did not capture state data.
- Administrators did not know how to use the tools to capture troubleshooting data.

You often end up with a system that has been restarted, no data to understand the failure state, and even worse, no way to begin troubleshooting until yet another outage has occurred.

Doing it Right

The best admin teams do things:

- All appropriate performance troubleshooting tools are in place and, admins are trained in how to use them. A good starting point is Atlassian’s tools at https://bitbucket.org/atlassianlabs/atlassian-support. You can use this script to capture thread dumps, heap dumps, and test both disk and database speeds.
- Garbage Collection (GC) logs, access logs at both the application and proxy level, and JMX monitoring are all in place.
- Database backups are being taken on a regular basis and validated as working.
- Atlas Load is permanently running, writing all recordings to a volume where space is not an issue. Recordings are pruned with the same regular schedule as Database backups are deleted.

If we reproduce our previous scenario with this environment, the admin would take a DB backup at a time before the outage, restore it in staging, grab the Atlas Load recording, and reproduce the failure. Logging can be changed as needed, new tools can be used, and even external analysts can be brought in to help. The key here is that now you can reproduce at will, you are no longer dependent on waiting for more outages in your production environment to continue investigating the root cause.
Application Trust

It's one thing to have an outage and go back to fix the problem, and it's another to ensure that it doesn't happen in the first place. This is where improved testing procedures support an improved level of trust in the tools.

Fixes

In the previous section, we identified a cause of your outages and wanted to deploy it ASAP, but how do you know that the fix you want to implement will help? If you push to production, when do you finally get to say that things are fixed? By validating a fix in your staging environment where you know the issue can be consistently reproduced you gain both the peace of mind that the issue is resolved, as well as the concrete timeline to fix validation.

Regression Avoidance

It's an application vendors responsibility to do high-quality regression testing, but as an application owner, you also hold responsibility for validating that your application is moving forward sustainably. There are many operations that require high-quality regression testing including, but not limited to:

- Application update
- Infrastructure changes
- App (plugin / add-on) update

Before deploying any of these in production, you should make sure nothing breaks. Using Atlas Load to validate the changes while monitoring both HTTP responses and application monitoring metrics gives you a much higher level of confidence before going to production.

Excessive Risk Mitigation

You're working with Atlassian support, and they tell you that they need you to delete data via the database. You do your due diligence and take a good backup before deleting anything. The deletion goes off without a hitch, and the next day, your users don't see anything wrong. A week later, trouble strikes, and you find out that you did cause irreparable damage. You can either roll back and lose a week of work, or you can try and massage the data back together via the DB; though this is often impossible.

If you have a recording of what happened after the deletion, all you need to do is restore your pre-deletion backup, then replay your traffic, and you're in a much better state than either of the earlier scenarios and with less work.
Data Center

You've worked with Atlassian Support and a Partner to tune the JVM and update the configs, but it's just getting to be a bit too much. The Partner, Atlassian Support, and yourself all agree the next step is Data Center. Now comes the tough part, explain to your boss how the $16,500 per year tool for your 10K user Jira license, is going to cost $240,000 per year instead. What evidence can you show your boss that increasing annual spend on tooling by 14.5x will solve the problem? Intuitively, we know spreading the load across nodes should help, but how much?

Now is when you bring in Atlas Load and validate that Data Center will help. Record your current performance, set up a Data Center evaluation, replay your traffic, and measure the impact that investment in Data Center will make.

A Better Baseline

New functionality is going to enter your environment, and you need to test this with your dataset. Situations include:

- New features introduced in an Atlassian application with an upgrade
- A new app (plugin/add-on) is added to your application
- Existing app (plugin/add-on) is updated to include new functionality

Atlas Load won’t be able to generate new traffic to test this new functionality. You will need to solve this with traditional tools like manual UAT, JMeter, Gatling, or E3. However, testing without a realistic traffic baseline for existing functionality is a recipe for inaccurate testing.

Value

In the end, the question comes down to ROI. The average burdened salary for a US software engineer is roughly $76,000. With a team of 500 engineers, an hour of outage comes at an opportunity cost $19,000. Slow performance in collaboration tools will also contribute to ongoing switching costs that slow your staff down.

Freebies - Audit Logging

If you run Atlas Load as a persistent recording with historical pruning synced with DB pruning, then you have the highest quality audit log possible. For organizations where Atlassian's integrated audit logging tools or proxy-based access logs do not suffice, this is an added benefit you get for free.