Common Agent Tasks

Namazu Agent · https://namazustudios.com/docs/namazu-agent/namazu-agent-common-tasks/

This page walks through the everyday workflow of working with the Namazu Agent: starting a session in Namazu Cloud, connecting it to GitHub, setting up your workspace, previewing changes on a temporary instance, and finally moving your game to production. It is the practical companion to the overview, which explains what the agent is and why you would use one.

The Development Workflow at a Glance

You and the agent work in a loop. You do the parts that need taste and judgment, the agent does the parts that need code slinging, and GitHub is the handoff point between you.

[Namazu Agent development workflow]

1. Start a session in Namazu Cloud
2. Authenticate with GitHub
3. Have the agent set up the workspace so it can push and pull your game's code from GitHub
4. Ask it to set up a temporary instance of Namazu Elements, a copy of your main production instance, to preview against
5. Iterate: the agent pushes code changes to GitHub for you, you pull them locally, add art, review, test, and adjust, then push back
6. When you are satisfied, ask the agent to move your game to production

Starting a Session

Begin from the Namazu Cloud control panel by launching an agent session. Under the hood this dispatches a job via Namazu Conductor, which starts the agent's execution environment in its own workspace and wires up the terminal you interact with in the dashboard. You do not need to provision

anything yourself; starting the session is all it takes.

First-Run Onboarding

On its very first run in a fresh workspace, the agent walks you through a short onboarding flow instead of dropping you into a blank prompt. Expect a handful of questions, such as whether you are starting a new or existing project, which game engine or client framework you're using, and where the project's source lives. Answer them, and the agent sets up accordingly.

Authenticate with GitHub

The agent pushes and pulls code for you, so it needs access to your GitHub account. It runs `gh auth login` as part of startup, and you complete the sign-in that it prompts you for, usually via a device code or a link to open. This is where the agent gets permission to clone your repos, commit, push, and open pull requests on your behalf. Because the running container is disposable (see Session Persistence below), you may need to sign in again at the start of a fresh session.

Set Up Your Workspace

Ask the agent to pull your game's code into the workspace, or to scaffold a new project, and it sets up the whole structure: the server-side Elements project, the client or game code, and the reference material it bases its work on. From then on the workspace is the agent's working copy, with GitHub as the bridge to your own machine.

Temporary Preview Instance

Before anything touches a real deployment, ask the agent to set up a temporary instance of Namazu Elements: a disposable preview running its own copy of your project's backend and database, separate from your main production instance. It gets its own externally browsable URL, so you and the agent can verify changes live without any risk to production. When you are done with it, the agent tears it down.

The Core Iteration Loop

Day to day, you and the agent move in a loop that keeps the actual game-making on your machine:

1. The agent writes or changes code and pushes it to GitHub for you
2. You pull that branch on your local machine
3. You add your art, review the code, test it locally, and make adjustments
4. You push your changes back to GitHub
5. The agent picks up the changes and continues

You never hand a zip over a chat window, and the agent never works on a different copy than you do. GitHub is the single shared source of truth between you.

Moving to Production

When you are satisfied with a change, ask the agent to move your game to production. The agent deploys the verified work to your real, running instance as a live upgrade. Confirm that is what you want explicitly for that specific change; the agent treats a deploy to production as a one-way door and only does it on your clear go-ahead, not on a general okay from earlier in the conversation.

Session Persistence

Two very different things survive between sessions:

- **The project workspace persists.** The workspace that holds your repos and working files is kept across separate sessions, so you do not have to rebuild your project each time you start a session.
- **The running container is fresh every time.** Each session runs in a new, disposable execution environment. Anything that lives only inside that container, such as a logged-in state stored in the container's own filesystem, does not survive between sessions. GitHub and other provider sign-ins are the practical case: you will usually sign in again at the start of each session rather than being silently carried over.

Idle Timeout and Notifications

A session that is dispatched but not actively used winds itself down automatically after a period of inactivity, so leaving a tab open does not leave the meter running. You can also stop a session manually from the control panel at any time. Each session is a per-job compute cost, not a standing service, so only start one when you actually need it.

While a session is running, the agent does not just print text and hope you are watching. When it finishes a long task, gets blocked on a decision only you can make, or needs to hand over a link or a credential, it surfaces a toast notification in the dashboard to get your attention. Keep an eye on those toasts; they are how the agent asks you to weigh in between its own turns.

Related Pages

- Namazu Agent: what the agent is, the agent variants, model options, and how sessions run
- Namazu Conductor: the orchestration engine that dispatches and manages agent sessions
- Deploying an Element: what a deployable Element looks like when you promote work to production