Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Migrating to SvelteKit 3

Class Duration

7 hours of live training delivered over 1 day.

Student Prerequisites

- Working experience with a SvelteKit 2 application
- Svelte 5 and runes experience is helpful
- Familiarity with Git, npm, and the team's test suite

Target Audience

Developers, tech leads, and platform teams responsible for SvelteKit 2 apps or shared Svelte libraries. The course works best when teams bring a real codebase to migrate. Teams that need a refresher on current Svelte can start with Svelte Essentials.

Description

SvelteKit 3 moves configuration into the Vite plugin, replaces `$lib` with `#lib`, and removes `$app/stores`. It also raises the minimums to Node.js 22.17, TypeScript 6, and Vite 8. Students first prepare on the latest 2.x release, then run `npx sv migrate sveltekit-3` and review each task's diff. They work through the remaining breaking changes area by area, from imports and errors to cookies and adapters. The course closes with clearing leftover TODOs with an AI coding agent, then testing and rolling out the upgrade.

Learning Outcomes

- Assess a SvelteKit 2 codebase for SvelteKit 3 readiness.
- Prepare on the latest 2.x release by fixing deprecation warnings.
- Run `npx sv migrate sveltekit-3` and review the resulting diff.
- Resolve each SvelteKit 3 breaking change, from configuration to adapters.
- Turn `@migration-task` comments and `svelte-check` errors into a task list.
- Clear the task list with an AI coding agent under human review.
- Test, stage, and roll out the upgrade with a rollback plan.

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

Students will need a free, personal GitHub account to access the courseware. Students need permission to install Node.js and Visual Studio Code on their computers. Also, students will need permission to install NPM Packages and Visual Studio Code extensions. An AI coding agent (such as Claude Code, GitHub Copilot, or Cursor) is recommended for the TODO list topics. We will provide a cloud-based environment if students cannot configure a local environment.

Software versions: SvelteKit 2 (latest 2.x) to SvelteKit 3, with Svelte 5, Vite 8, TypeScript 6, and Node.js 22.17 or later.

Training Topics

Planning the Upgrade

- What changed in SvelteKit 3
- Node.js 22.17 and TypeScript 6 minimums
- Vite 8 with Rolldown
- `@sveltejs/vite-plugin-svelte` 7
- Svelte 5 as a requirement
- Converting remaining Svelte 4 code
- Auditing adapters, plugins, and libraries
- Inventorying config, imports, and stores
- Environment variables and service workers
- A baseline of tests, `svelte-check`, and builds

Preparing on the Latest 2.x

- Upgrading to the latest SvelteKit 2
- Fixing targeted deprecation warnings
- Moving config into `vite.config.ts` on 2.x
- Adopting `$app/state` with `npx sv migrate app-state`
- Trying explicit environment variables
- `resolve()` and `asset()` in place of `base`
- Committing preparation work separately

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Running `npx sv migrate sveltekit-3`

- Reading the changelog first
- Running every task with `--tasks all`
- Running one task at a time
- The `package-json` and `tsconfig` prerequisites
- The code tasks, from `svelte-config` to `app-state`
- What the migration leaves for you
- Reviewing the diff
- Finding `@migration-task` comments

Configuration

- From `svelte.config.js` to `vite.config.ts`
- `sveltekit()` plugin options
- `compilerOptions` beside SvelteKit options
- Removed options such as `files.lib`
- Removed `prerender.origin` and `csrf.checkOrigin`
- New `csrf.trustedOrigins` and `paths.origin`
- `output.linkHeaderPreload`
- The new `version.pollInterval` default

Imports and TypeScript

- `$lib` to `#lib`
- `package.json` `imports` and file extensions
- Extending `$app/tsconfig`
- Explicit `include` and `exclude`
- `$app/environment` renamed to `$app/env`
- From `$env/*` to `src/env.ts`
- `$app/env/private` and `$app/env/public`
- Relocated hook, param, and env types
- Remote types in `$app/server`

State and Navigation

- `$app/stores` removed
- From `$page` to `page` in `$app/state`
- Read-only `page.url`
- `updated.current` version checks
- `goto` with `shallow` in place of `pushState`

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

- `invalidateAll` to `refreshAll`
- Renamed `goto` options
- `goto` rejecting unknown routes
- Error results from `preloadData`
- `use:enhance` with actions on other pages

Paths

- `base`, `assets`, and `resolveRoute` removed
- `asset()` and `resolve()` without leading slashes
- `Path` and `AssetPath` types

Errors

- `status` on every `App.Error`
- String messages in `error()`
- Extra properties in a third argument
- Validation errors in `handleError`
- Setting status codes from `handleError`
- Render errors and error boundaries
- `fail()` status codes in enhanced forms

Routing, Responses, and Server Code

- Param matchers in `src/params.ts`
- Matchers with `defineParams`
- Server-only `server` file segments
- Empty 2xx responses
- `resolve()` in `handle` returning a promise
- `Response` in place of `json()` and `text()`
- External redirects with `external: true`
- `data-sveltekit-*` values of `false`
- Same-page links and `refreshAll()`
- Universal `config` overriding server `config`

Security and Cookies

- Always-on CSRF protection
- `csrf.trustedOrigins`
- Cross-origin requests without `Content-Type`
- ASCII cookie names
- The default `/` cookie path

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

- Renamed cookie option types

Service Workers and Observability

- `$service-worker` removed
- `$app/env`, `$app/manifest`, and `$app/paths` in workers
- Typed workers with `$app/service-worker`
- A separate service worker tsconfig
- Module service workers
- Automatic `instrumentation.server` loading
- Opt-in OpenTelemetry tracing

Adapters

- `adapter-node` static asset lists
- GET and HEAD only for static files
- Content-hash ETags
- `adapter-cloudflare` and `cloudflare:workers`
- `adapter-netlify` changes
- `adapter-vercel` without the edge runtime
- Synchronous `getRequest` and `setResponse`
- The removed polyfills module
- Custom adapters and the Builder API

Remote Functions (Experimental)

- Enabling remote functions in the Vite plugin
- Enabling async Svelte
- The `remote` file name segment
- Form fields with `field.as()`
- Request data no longer available in queries

Clearing the TODO List with an AI Coding Agent

- Collecting TODOs into one task list
- `@migration-task` comments and `svelte-check` errors
- Failing tests as tasks
- Grounding agents with the Svelte MCP server
- `llms.txt` documentation files
- Agent instructions for a migration
- Checking code with `svelte-autofixer`
- One change category per commit

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Reviewing agent diffs

Testing and Rollout

- Re-running tests against the baseline
- Comparing builds, bundles, and server behavior
- Adapter-specific checks
- Staging and preview deployments
- Migrating libraries built with `svelte-package`
- Rolling out across apps and monorepos
- Rollback plans and monitoring
- Updating conventions, lint rules, and templates

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved