Serverless Framework Deploy Lambda from YAML

START HERE

`serverless login`
Sign in to the Framework

`serverless info`
Show stage, endpoints, functions

`serverless deploy`
Deploy the whole service

`serverless remove`
Delete the stack

`serverless deploy function -f <fn>`
Fast single-function deploy

`serverless invoke local -f <fn>`
Run a function locally

`serverless invoke -f <fn> --log`
Call the deployed function

`serverless logs -f <fn> -t`
Tail CloudWatch logs

serverless.yml functions + events → `package` → **.serverless/** zips + CloudFormation → `deploy` → **CloudFormation stack** Lambda, API, IAM → `invoke` / `logs` → **Running service** per stage and region

CONCEPTS

sls is a short alias for serverless. Every command reads serverless.yml in the current folder and defaults to stage dev.

A stage (--stage) is a separate full copy of the stack, e.g. dev, staging, prod. Region comes from --region or provider.region.

Add --aws-profile <name> to any command to use a named AWS profile instead of the default credentials.

ACCOUNT

`serverless login`
Sign in and pick the default org

`serverless login --org <name>`
Switch the default org

`serverless <command> --help`
Show help for one command
Works without signing in

DEPLOY

`serverless deploy`
Deploy the whole service

`serverless deploy --stage <stage> --region <region>`
Deploy to another stage or region

DEPLOY (cont.)

`serverless deploy --verbose`
Show stack events and outputs

`serverless deploy --force`
Force a deployment

`serverless deploy --aws-profile <name>`
Deploy with a named AWS profile

`serverless deploy --package <dir>`
Deploy a pre-built package

`serverless deploy --conceal`
Hide secrets in the output

`serverless deploy function -f <fn>`
Upload one function's code only
Skips CloudFormation, so it is fast; not for event or IAM changes

`serverless deploy function -f <fn> --update-config`
Push only Lambda config changes

`serverless deploy list`
List stored deployments

`serverless deploy list functions`
List deployed function versions

PACKAGE

`serverless package`
Build artifacts without deploying

`serverless package --package <dir>`
Write the package to a folder

INVOKE

`serverless invoke -f <fn>`
Call the deployed function

`serverless invoke -f <fn> --log`
Also print the invocation log

`serverless invoke -f <fn> -d '<json>'`
Pass an inline event

`serverless invoke -f <fn> -p <file>`
Pass an event from a JSON file

`serverless invoke -f <fn> -t Event`
Fire asynchronously
Types: RequestResponse (default), Event, DryRun

LOCAL

`serverless invoke local -f <fn>`
Run a function on your machine

`serverless invoke local -f <fn> -p <file>`
Use an event file

LOCAL (cont.)

`serverless invoke local -f <fn> -d '<json>'`
Use an inline event

`serverless invoke local -f <fn> -e <NAME>=<value>`
Set an environment variable

`serverless invoke local -f <fn> --docker`
Run in the declared runtime
Without it, Node.js and Python use your local interpreter

`serverless dev`
Run deployed functions on local code
Attaches Node.js/Python functions of a deployed stage; refuses prod

LOGS

`serverless logs -f <fn>`
Show the last 10 minutes of logs

`serverless logs -f <fn> -t`
Tail logs live

`serverless logs -f <fn> --startTime <time>`
Logs since 30m, 2h, 3d or a date

`serverless logs -f <fn> --filter <text>`
Only lines containing a pattern

`serverless metrics`
Service-wide metrics, last 24 hours

LOGS (cont.)

`serverless metrics -f <fn> --startTime <t>`
Metrics for one function

INSPECT

`serverless info`
Show endpoints and functions

`serverless info --verbose`
Include all stack outputs

`serverless print`
Show config with variables resolved

`serverless print --path <a.b>`
Print one sub-value

`serverless print --format json`
Output as yaml, json or text

ROLLBACK AND REMOVE

`serverless rollback`
List deployments to roll back to

`serverless rollback -t <timestamp>`
Return to a past deployment
Timestamps come from serverless deploy list

`serverless remove`
Delete the service from AWS

`serverless remove --stage <stage> --region <region>`
Delete one stage

CONFIG

Minimal serverless.yml

```
service: my-api

provider:
  name: aws
  runtime: nodejs20.x
  region: eu-central-1

functions:
  hello:
    handler: handler.hello
    events:
      - httpApi: GET /hello
```

Use ${opt:stage, 'dev'} to read a CLI option, ${env:NAME} for environment variables, ${self:custom.x} for other keys.

Set org: <name> in serverless.yml to bind a service to an org; SERVERLESS_ACCESS_KEY signs in non-interactively in CI.

RECIPES

Iterate on one function

```
# edit handler.js, then
serverless deploy function -f hello
serverless invoke -f hello --log
serverless logs -f hello -t
```