DFX Cheatsheet for the Internet Computer

---

1. Project & Environment

dfx new <project>

- **Creates** a new DFX project with basic structure.
- **Common flags**
  - `--frontend <framework>`: React, Vue, Svelte, etc.
  - `--type <language>`: Motoko, Rust, Azle, Kybra
  - `--dry-run`, `--extras`
- **Example**

```
dfx new hello_ic cd hello_ic
```

dfx start / dfx stop

- **dfx start**: Spins up a local replica (IC emulator).
  - `--background`: Runs in background
  - `--clean`: Clears old state
  - `--no-artificial-delay`: Disables artificial latency
- **dfx stop**: Stops the local environment.
- **Example**

```
dfx start --clean --background dfx stop
```

dfx build / dfx deploy

- **dfx build**: Compiles canisters from your project.
- **dfx deploy**: Registers and installs compiled code to a target environment (local or `--network ic`).
- **Example**

```
dfx build dfx deploy dfx deploy my_canister --network ic --argument '(42)'
```

dfx help / dfx version / dfx cache

- **dfx help [subcommand]**: Quick usage info.
- **dfx version**: Shows current DFX version.
- **dfx cache**
  - `dfx cache list`: Lists cached DFX versions
  - `dfx cache install <ver>`: Installs a specific version
  - `dfx cache delete <ver>`: Removes a cached version

---

2. Canister Management

dfx canister create <name>

- Registers a **new, empty** canister.
- **Key options**: `--memory-allocation`, `--compute-allocation`, `--with-cycles`
- **Example**

```
dfx canister create my_canister --with-cycles 1000000000
```

dfx canister install <name>

- Installs compiled Wasm into the canister.
- **Key options**
  - `--mode install | reinstall | upgrade | auto`
  - `--argument <data>`, `--argument-file <file>`
  - `-y` (auto-confirm)
- **Example**

```
dfx canister install my_canister --mode upgrade
```

dfx canister uninstall-code <name>

- Removes the code & state but **retains** the canister ID.

dfx canister stop <name> / dfx canister start <name>

- Manually stops/starts the execution of a canister.

dfx canister delete <name>

- Permanently **deletes** a canister (must be stopped first).

dfx canister call <name> <method> [args]

- Invokes a method on the deployed canister.

```
dfx canister call my_canister greet '("Hello World")'
```

dfx canister status <name>

- Shows canister details (cycle balance, memory, controller).

dfx canister id <name>

- Displays the canister's unique ID.

dfx canister info <name> / dfx canister metadata <name> <field>

- Provides deeper info (Wasm hashes, metadata fields, etc.).

dfx canister update-settings <name> [options]

- Updates settings like controller or allocations.

dfx canister deposit-cycles / withdraw-cycles

- `deposit-cycles`: Transfer cycles **into** a canister.
- `withdraw-cycles`: Transfer cycles **out** of a canister.

dfx canister logs <name>

- Retrieves logs from a running canister.

---

3. Identity & Cycles

dfx identity new <id>

- Creates a new **local identity**.
  - `--storage-mode <mode>`, `--force`

dfx identity use <id>

- Switches current identity context.

dfx identity list / whoami

- Lists all identities (current one marked with `*`) / prints current identity name.

dfx identity get-principal / get-wallet

- Displays the **principal** or wallet canister for active identity.

dfx identity set-wallet <canister_id>

- Manually sets wallet canister for active identity.

dfx identity deploy-wallet <canister_id>

- Installs the wallet Wasm code into a chosen canister.

dfx identity export/import <id>

- Exports/imports private keys to/from PEM files.

dfx cycles balance / top-up / convert / transfer / approve

- **balance**: Shows cycle balance (identity or principal).
- **top-up**: Adds cycles to a canister.
- **convert**: Converts ICP to cycles.
- **transfer**: Sends cycles to another principal.
- **approve**: Grants a principal spending authority.

---

4. Network & Deployment

dfx deploy [<canister>] [options]

- Deploys canisters to **local** or a specified network (`--network ic`).
- **Modes**: `install`, `reinstall`, `upgrade`, `auto`

dfx ping [network]

- Tests connectivity to a given network.

dfx network [subcommand]

- Manages custom network definitions (beyond local/ic).

dfx deps pull/init/deploy

- Handles external canister dependencies declared in `dfx.json`.

dfx generate <canister>

- Generates type declarations / boilerplate (JavaScript, TypeScript, Candid, etc.).

---

5. Wallet & Ledger (Optional or Advanced)

dfx wallet balance/send [--network <net>]

- `wallet balance`: Cycle balance in your wallet.
- `wallet send <dst_canister> <amount>`: Sends cycles from your wallet to a canister.

dfx wallet authorize/deauthorize/add-controller/remove-controller

- Manage who controls / can spend from your wallet canister.

dfx wallet upgrade / redeem-faucet-coupon

- Upgrades wallet's Wasm code / redeems a dev "faucet coupon" for cycles.

dfx ledger account-id / balance / transfer / create-canister / top-up

- **ledger account-id**: Prints an account ID (default identity or a principal).
- **ledger transfer**: Moves ICP to another account ID.
- **ledger create-canister/top-up**: Creates or funds a canister by converting ICP to cycles.

---

6. NNS & SNS Extensions (Optional or Advanced)

- **dfx extension install nns** / **dfx nns [import/install/help]**
  For interacting with the **Network Nervous System** canisters (e.g., governance, ledger, root).
- **dfx extension install sns** / **dfx sns [create/validate/deploy/help]**
  For deploying and managing a **Service Nervous System** (decentralized governance for your dapp).

Cheatsheet Tips

- `--network <network>`: Almost any deploy or canister command can target either `local` or `ic` (mainnet).
- `dfx help <command>`: Always your first reference for usage details.
- `dfx version`: Verify your DFX version to ensure compatibility.
- **Project file**: `dfx.json` is crucial—holds canister definitions, network settings, and more.
- **Experiment locally**: Use `dfx start` to test your code cost-free before hitting mainnet.

Further Reading

For in-depth documentation and the latest updates, consult:
internetcomputer.org/docs

Happy building on the Internet Computer!
