Infrastructure as code for MikroTik routers



Scripting the setup of new MikroTik devices with the RouterOS deck: desired state in a file, a plan, a dry run, backup, apply, verify, roll back.

Based on **Jennifer 0.26.0**, November 2026, by @mplx.

Why infrastructure as code

- **Clicking in WinBox** works for one router. For thirty branch offices it means thirty slightly different configurations, and nobody knows which.
- **A file per device** in a repository: reviewed, diffed, versioned.
- **A program** that reads the router, compares, and changes only what differs: run it again and nothing happens.
- **Same steps every time:** dry run, backup, apply, verify, roll back on error.

The decks

Deck	What it is
@deckyard/routeros	the thick layer: typed verbs for 56 topics, input validated before sending
@jennifer/mikrotik	the wire client: sessions, sentences, tags, /listen streams

```
jvc add @deckyard/routeros   # into vendor/deckyard/routeros/

import "@deckyard/routeros/" as mt;
```

Desired state: devices/branch-07.toml

```
name = "branch-07"
host = "10.7.0.1"
port = 8729
tls = true
identity = "branch-07"
dns = "9.9.9.9,149.112.112.112"
[[item]]
path = "/ip/address"
comment = "iac:lan"
address = "10.7.0.1/24"
interface = "bridge"
```

Loading it

```
def doc as toml.Value init toml.decode(fs.readString($file));
for (def i in 0..toml.length($doc, "/item")) {
    def at as string init "/item/{$i}";
    def it as Item init Item{
        path: toml.asString($doc, $at + "/path"),
        comment: toml.asString($doc, $at + "/comment"),
        attrs: {}
    };
    for (def k in toml.keys($doc, $at)) {
        if ($k != "path") {
            $it.attrs[$k] = toml.asString($doc, $at + "/" + $k);
        }
    }
```

Connecting: api or api-ssl, secrets from the environment

```
def c as mt.Client init mt.connectWith(
    $want.host,
    $want.port,
    os.getEnv("MT_USER"),
    os.getEnv("MT_PASSWORD"),
    $want.tls);
defer mt.disconnect($c);
```

- `mt.connectTLS(host, user, password)`: api-ssl on 8729, certificate verified
- `mt.connect(...)`: plain API on 8728, the login in clear text: lab only
- a dedicated RouterOS user for the program, its password never in the repository

Read, then compute the difference

```
func changed(want as map of string to string, row as map of string to
string) {
    def out as map of string to string;
    for (def k in $want) {
        if (not maps.has($row, $k) or $row[$k] != $want[$k]) {
            $out[$k] = $want[$k];
        }
    }
    return $out;
}
```

Per path: wanted but missing gives **add**, present but different gives **set** (only the changed attributes), owned (`iac:`) but no longer wanted gives **remove**.

Dry run: the plan, nothing changed

```
$ jennifer run converge.j --dry-run
branch-07  ~ identity MikroTik -> branch-07
branch-07  ~ dns 8.8.8.8 -> 9.9.9.9,149.112.112.112
branch-07  ~ /ip/address iac:lan address=10.7.0.1/24
branch-07  + /ip/firewall/address-list iac:mgmt list=mgmt address=10.0.
0.0/24
branch-07  - /ip/firewall/address-list iac:old-office
branch-08  in sync
branch-09  ~ identity MikroTik -> branch-09
branch-09  + /ip/address iac:lan address=10.9.0.1/24 interface=ether9
branch-10  unreachable: refused
```

Apply: backup first, keep an undo list

```
def pw as string init os.getEnv("MT_BACKUP_PASSWORD");
mt.saveBackupWith($c, "iac-" + $stamp, $pw);
apply($c, $changes);
if (len(plan($want, readState($c, $want))) > 0) {
    throw Error{kind: "value", message: "not converged"};
}

try {                                   # in apply()
    for (def ch in $changes) { $undo[] = applyOne($c, $ch); }
} catch (e) {
    for (def u in lists.reverse($undo)) { applyOne($c, $u); }
    throw $e;
}
```

Rollback in action

```
$ MT_VERBOSE=1 jennifer run converge.j
mt> /system/backup/save name=iac-20261116-225017 password=<redacted>
mt> /file/print
mt< 1 row
mt> /system/identity/set name=branch-09
mt> /ip/dns/set servers=9.9.9.9,149.112.112.112
mt> /ip/address/add comment=iac:lan address=10.9.0.1/24
interface=ether9
mt> /ip/dns/set servers=8.8.8.8
mt> /system/identity/set name=MikroTik
branch-09  refused, rolled back: input does not match any value of
interface
```

Errors are typed structs

```
try {                                       # in runDevice()
    $r.lines = converge(load($file), $dryRun);
    $r.ok = true;
} catch (e) {
    if ($e.kind == "mikrotik") {            # mikrotik.MikrotikError
        $r.lines = ["refused, rolled back: " + $e.routerMessage];
    } elseif ($e.kind == "net") {           # net.NetError
        $r.lines = ["unreachable: " + $e.reason];
    } else {
        $r.lines = [$e.kind + ": " + $e.message];
    }
}
```

Many devices at once, at most four

```
for (def f in $files) { channel.send($jobs, $f); }
channel.close($jobs);
for (def n in 0..LIMIT) {
    $workers[] = spawn { return worker($jobs, $dryRun); };
}

while (true) {                              # in worker()
    try {
        $out[] = runDevice(channel.recv($jobs), $dryRun);
    } catch (e) {
        if ($e.kind != "closed") { throw $e; }
        return $out;
    }
}
```

Run it, then run it again

```
$ jennifer run converge.j; echo "exit $?"
branch-07  ~ /ip/address iac:lan address=10.7.0.1/24
branch-07  applied 5 change(s), verified
branch-08  in sync
branch-09  refused, rolled back: input does not match any value of
interface
branch-10  unreachable: refused
exit 1
$ jennifer run converge.j
branch-07  in sync
branch-08  in sync
```

Take-aways and read more

- Desired state in files, secrets in the environment, ownership by comment
- Read, plan, dry run, back up, apply with undo, verify: the same every time
- Branch on the error's kind, read its fields
- The deck: deckyard/deck-routeros, the wire client: jennifer-language/deck-mikrotik
- The language: jennifer-lang.dev, this talk in the Jennifer Academy

Based on Jennifer 0.26.0, November 2026.