Building in zones

> **Deprecation warning**
>
> This page is mostly about creating zone, so relevant content has been migrated to https://docs.openindiana.org/handbook/systems-administration/#zones . Avoid touching this wiki page.

Creating a zone for building software

> *Local zones are a lightweight virtualization technology, one of many goodies introduced by Sun Solaris 10 and OpenSolaris – a separate operating environment which runs within your host Solaris-based OS and has a separate set of installed software, so that jobs executed inside it – like the builds and subsequent tests have little impact on your main host OS, or on other local zones. For users coming from other environments, zones may look similar to "chroot jails" with kernel-augmented separation of user and process namespaces. Using ZFS, it is easy to snapshot the zone's filesystems, and roll back in case of major mishaps.*
> *Being an operating environment, a local zone may have a networking setup of its own – but a dedicated IP address number from your LAN is not required. The local zone may be NATed by its global zone using "etherstubs" (another goodie introduced in OpenSolaris) and other virtual networking components, or it may have no networking at all – but repository cloning and other networked operations would have to be run from the global zone then.*

We recommend building all software inside local zones, so you can set up a clean development environment and not "pollute" your main operating environment with your test packages which are not guaranteed to always work as expected 👷

It may make sense to have several build zones, say one for each major project, or using a "pristine" one for testing of your packaged builds when you think your project is nearing the finish line. In these cases it is recommended to set up one build zone properly, take its datasets' ZFS snapshot, and then clone it and reconfigure the clones as new local zones – this would guarantee that the zones are identical and save you some disk space and internet traffic.

In particular, local zones can be used along with multiple cloned workspaces (see Working on several bugs at once) for preconfigured compilation of the same project code-base with different supported compiler suites (SunStudio, GCC, whatever comes next?)

> **Distro Constructor**
>
> The Distribution Constructor (the software that produces the final ISO Image) does not work inside a local zone.

About local zone networking

OpenSolaris' (and its descendants') local zones can have at least two approaches to networking: a **shared IP stack** and an **exclusive IP stack**. There are pros and cons to each of them – they are tools for different jobs somewhat.

Low-level system networking components, such as the `ipfilter` firewall and the kernel IP routing tables attach to an "IP stack", and are thus either unique to a zone or shared by all zones with the one shared stack (usually including the global zone). There are also some other nuances, such as that the zones with the shared stack can communicate over IP directly, regardless of their subnetting and, to some extent, default firewall packet filtering (that has to be specially configured), while exclusive-IP zones with addresses in different subnets have to communicate over external routers and are subject to common firewall filtering.

It is the global zone however that defines *which* physical networks and VLANs the local zone has access to, and hands down the predefined networking interfaces (the local zone can not use or create other interfaces). Also, while shared networking allows to configure and attach (or detach) network interfaces from the global zone to the local zone "on the fly", changes in exclusive networking require reboot of the zone to propagate device delegation.

A local zone with an exclusive IP stack can have most or all the benefits of dedicated hardware networking, including a firewall, access to promiscuous sniffing, routing, configuration of its own IP address (including use of DHCP and static network addressing), etc. This requires a fully dedicated NIC however, which was a limitation in early OpenSolaris, until the CrossBow project came along bringing the ability to create VNICs, which are virtual adapters with their own MAC addresses, that operate as if the VNIC was plugged directly into your local LAN in a particular (or default) VLAN.
Note also that the local zones with an exclusive IP stack are **not** subject to the host's shared-stack firewall, and so are by default open to the external network with all possible security holes. You're encouraged to revise and minimize the potential breach exposure by disabling unneeded services as well as by configuring at least basic firewall rules in the zone (not covered in this manual).

It is also possible to use "shared networking" where the local zone's interface would be an alias of a NIC available to the global zone (a guide mentioning this is available here: HOW-TO Setup referential build zone for OpenIndiana Addon Consolidations) and the routing tables would be a subset of those available to the global zone.

If your build machine is a VM on a platform which requires the hypervisor host to "know" the VM's MAC addresses – such as VirtualBox running on Solaris with the "bridged networking" mode – your build zone must use a NIC defined by "VM hardware". You still have two options though:

- Use shared networking, attaching local zones to the NIC used by your global zone;
- Create a secondary NIC for your VM and dedicate it to an exclusive-IP zone (making a new VM NIC for each such zone).

If you plan to use VirtualBox bridged networking on a VM under Solaris-related OSes, see the VirtualBox User Guide for more details on this setup – in particular, the host Solaris machine is also encouraged to use VNICs with explicitly defined MAC addresses in order to attach the VM NICs to them.

Set up host resources for the zone

Create a zfs filesystem as a container for build-zones

The example below creates the zone root under `rpool` ZFS pool.
If your machine has other pools, perhaps bigger and/or more performant i.e. by using L2ARC caches, you may want to use a different pool for zone data (by further delegating whole datasets or `lofs`-mounting individual paths – such as your building workspaces), or for the whole zone roots altogether. As an alternative to loopback-mounting, the NFS client in a local zone can use the global zone's NFS server, but that is likely to be slow for compilation in particular (especially if `sync` is enabled without a fast ZIL).
(Note that it may be officially unsupported to hold local zone roots separately from OS roots in some OpenSolaris descendants; as well as that earlier OpenSolaris releases officially disapproved of local zones using their host global zone's NFS server – although it "just worked").

In this example we will host all zones for building inside a dedicated ZFS dataset mounted at `/zones/build` (each local zone will have its datasets hanging under this point). On one hand this allows you to group zones with different tasks for administrative visibility, on the other – the common container dataset can be used to inherit specific ZFS properties to the local zones. The container has some explicitly configured ZFS properties, but holds no data on its own (other than automatically created directories as mountpoints for child datasets).

Here, we are disabling `atime` and `sync` to speed up builds. Note that setting `sync=disabled` may result in data loss in a power loss/system crash scenario, so only enable it for your build environment if you don't mind losing data (from recent writes just before the crash) – and there are few cases where it is not unrecommended to enable this feature:

```
$ pfexec zfs create -o compression=on -o mountpoint=/zones rpool/zones
$ pfexec zfs create -o sync=disabled -o atime=off rpool/zones/build
```

In the example above, we create the common container for local zones in `rpool/zones`, set its mountpoint to `/zones` in the host's common filesystem tree, and enable generic lightweight compression (`lzjb` by default). Afterwards we create the container for build zones with specific ZFS properties which are not generally recommended for other use-cases. The `rpool/zones/build` container inherits other ZFS properties from its ancestors (up to `rpool`), including compression and base mountpoint from its parent – so it will be automatically mounted as `/zones/build` unless you override that explicitly at some point of the hierarchy.

The united ZFS properties will be inherited down the road to actual local zone datasets hosted under this container dataset.

Create a crossbow VNIC

This guide below assumes you are on a LAN with DHCP, so we will create a CrossBow VNIC on your primary network interface (assumed to be `e1000g0` here, adapt as necessary). You can skip this step if the local zone would use shared networking or a dedicated (VM-)hardware NIC.

If your primary interface is not `e1000g0`, please substitute for the one that is in use. You can normally find this via "`dladm show-phys`" or "`ifconfig -a`".

```
$ pfexec dladm create-vnic -l e1000g0 vnic0
```

If needed, you can also specify binding of the VNIC to a particular VLAN of your external network with `-v` parameter.

Basically set up the local zone

Create the zone configuration

zonecfg for zones with an exclusive IP stack

```
$ pfexec zonecfg -z zone1
create
set autoboot=true
set zonepath=/zones/build/zone1
set ip-type=exclusive
add net
set physical=vnic0
end
exit
```

zonecfg for zones with a shared IP stack

Here the global zone predefines and fixes the IP addressing used by the local zone:

```
$ pfexec zonecfg -z zone1
create
set autoboot=true
set zonepath=/zones/build/zone1
set ip-type=shared
add net
set address=192.168.1.181/24
set physical=e1000g0
set defrouter=192.168.1.1
end
exit
```

Install the zone

NOTE: It may be possible to save on internet traffic by installing from a locally configured mirror of OpenIndiana IPS repositories set as defaults for your system. This is not explored below, but you're welcome to try that and write a howto chapter here 🚁

This step will install the zone by downloading packages from the internet (about 150Mb in this example):

```
$ pfexec zoneadm -z zone1 install
A ZFS file system has been created for this zone.
   Publisher: Using openindiana.org (http://pkg.openindiana.org/dev/ ).
       Image: Preparing at /zones/build/zone1/root.
       Cache: Using /var/pkg/publisher.
Sanity Check: Looking for 'entire' incorporation.
  Installing: Packages (output follows)
          Packages to install: 136
       Create boot environment:  No
Create backup boot environment:  No
            Services to change:   4

DOWNLOAD                                  PKGS       FILES    XFER (MB)
Completed                               136/136 28050/28050  149.9/149.9

PHASE                                       ACTIONS
Install Phase                             41423/41423

PHASE                                         ITEMS
Package State Update Phase                  136/136
Image State Update Phase                        2/2

        Note: Man pages can be obtained by installing pkg:/system/manual
 Postinstall: Copying SMF seed repository ... done.
 Postinstall: Applying workarounds.
        Done: Installation completed in 267.072 seconds.

  Next Steps: Boot the zone, then log into the zone console (zlogin -C)
              to complete the configuration process.
```

Troubleshooting

Zone creation as described above can fail for a number of reasons, sometimes obscure (i.e. you followed different steps, or your current OS deviates from one used in making this how-to).

One common problem is inability to create a dataset with the requested `zonepath` with error reports similar to this:

```
$ pfexec zoneadm -z zone1 install
ERROR: the zonepath must be a ZFS dataset.
The parent directory of the zonepath must be a ZFS dataset so that the
zonepath ZFS dataset can be created properly.
```

In this case an empty dataset with needed mountpoint has to be created manually.

This step can lead to the second common problem: zone root's filesystem access rights must only allow `root`:

```
$ ls -ld /zones/build/zone1
drwxr-xr-x   2 root     root           2 May  5 13:53 /zones/build/zone1

$ pfexec zoneadm -z zone1 install
/zones/build/zone1 must not be group readable.
/zones/build/zone1 must not be group executable.
/zones/build/zone1 must not be world readable.
/zones/build/zone1 must not be world executable.
could not verify zonepath /zones/build/zone1 because of the above errors.
zoneadm: zone zone1 failed to verify
```

These two problems can be amended with the following commands (after creation of zones container dataset as detailed above):

```
$ pfexec zfs create rpool/zones/build/zone1
$ pfexec chmod 700 /zones/build/zone1
```

Check the commands' output for errors, such as inability to mount the created dataset (before the `chmod` call).

Review that zone installation succeeded (optional👒)

This should automatically create and populate the local zone's dataset hierarchy:

```
$ pfexec zfs list -o name,used,avail,refer,compressratio,compression,mountpoint -r rpool/zones
NAME                              USED  AVAIL  REFER  RATIO  COMPRESS  MOUNTPOINT
rpool/zones                       285M  45.5G    32K  2.00x        on  /zones
rpool/zones/build                 285M  45.5G    32K  2.00x        on  /zones/build
rpool/zones/build/zone1           285M  45.5G    32K  2.00x        on  /zones/build/zone1
rpool/zones/build/zone1/ROOT      285M  45.5G    31K  2.00x        on  legacy
rpool/zones/build/zone1/ROOT/zbe  285M  45.5G   285M  2.00x        on  legacy
```

You can review the zone configuration by supported commands, as well as by accessing its XML-file descriptor (unsupported although convenient – subject to change in future versions):

```
### Review defined zones
$ pfexec zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              ipkg     shared
   - zone1            installed  /zones/build/zone1             ipkg     excl

### Review zone config in the way good for cloning
$ pfexec zonecfg -z zone1 export
create -b
set zonepath=/zones/build/zone1
set brand=ipkg
set autoboot=true
set ip-type=exclusive
add net
set physical=vnic0
end


###
### UNSUPPORTED methods follow
###
$ pfexec grep zone1 /etc/zones/index
zone1:installed:/zones/build/zone1:4a79102b-8029-67e7-b394-c8d4000d5950

$ pfexec cat /etc/zones/zone1.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE zone PUBLIC "-//Sun Microsystems Inc//DTD Zones//EN" "file:///usr/share/lib/xml/dtd/zonecfg.dtd.1">
<!--
    DO NOT EDIT THIS FILE.  Use zonecfg(1M) instead.
-->
<zone name="zone1" zonepath="/zones/build/zone1" autoboot="true" brand="ipkg" ip-type="exclusive">
  <network physical="vnic0"/>
</zone>
```

Snapshot the zone

You might want to clone another zone later with these basic packages, or roll back to current conditions:

```
$ pfexec zfs snapshot -r rpool/zones/build/zone1@initialPackages
```

Set up the local zone operating environment

Create a sysidcfg file (OBSOLETE: sysidcfg was superseded by sysding)

This step answers questions you would otherwise have to answer manually via a console-based wizard upon the first boot. Now the wizard would get its answers from the file (it may still ask questions not covered in the file, i.e. if the future versions of the wizard define new questions).

Mount the zone's ZFS dataset so we can access it by running:

```
$ pfexec zoneadm -z zone1 ready
```

sysidcfg for zones with an exclusive IP stack

Now create the `sysidcfg` file:

```
$ pfexec cat <<EOF > /zones/build/zone1/root/etc/sysidcfg
terminal=xterms
network_interface=PRIMARY {dhcp protocol_ipv6=no}
security_policy=none
name_service=NONE
nfs4_domain=dynamic
timezone=UTC
root_password=fto/dU8MKwQRI
EOF
```

Remark: the encrypted root password shown here is: `abc123`

Note also that in the example above, your zone will try to receive networking settings via DHCP. It is possible to set static IP addressing for an "`ip-type=exclusive`" local zone by using traditional Solaris methods with files (relative to zone root): `/etc/defaultrouter`, `/etc/hostname.vnic0`, `/etc/netmasks` and so on; in this case you might want to disable `nwam` and `dhcp-client` in the zone. You may also want to make sure that in the zone's `/etc/hosts` file the static IP address would be associated with the zone's network name (short hostname and full FQDN), and that name should be used in `/etc/hostname.vnic0` instead of an IP address directly.

You can also configure individual `ipfilter` firewall in the "`ip-type=exclusive`" zone (GZ's firewall doesn't apply to non-shared LZ networking).

sysidcfg for zones with a shared IP stack

Now create the `sysidcfg` file:

```
$ pfexec cat <<EOF > /zones/build/zone1/root/etc/sysidcfg
terminal=xterms
name_service=DNS { domain_name=myhome.com
     name_server=192.168.1.1,8.8.8.8
     search=myhome.com,mywork.com}
network_interface=PRIMARY {
     default_route=192.168.1.1
}
security_policy=none
nfs4_domain=dynamic
timezone=UTC
system_locale=C
timeserver=localhost
root_password=fto/dU8MKwQRI
EOF
```

Remark: the encrypted root password shown here is: `abc123`

After boot you may also want to make sure that in the zone's `/etc/hosts` file the static IP address (set by the global zone) would be associated with the zone's network name (short hostname and full FQDN).

Boot your zone

```
$ pfexec zoneadm -z zone1 boot
```

You will now want to attach to the zone's console and watch it boot, and answer any questions if prompted:

```
$ pfexec zlogin -C zone1
```

You can detach from the console by issuing "`~.`" without quotes (or "`~~.`" over chained Unix `ssh`).

DNS configuration in the zone

Once the zone has booted, you can copy the DNS resolution settings from GZ into the local zone (if networking is the same, and if your `sysidcfg` or DHCP setups did not take care of that successfully):

```
$ pfexec cp /etc/resolv.conf /etc/nsswitch.conf /zones/build/zone1/root/etc/
```

Final checks

You can now `zlogin` into the local zone with:

```
$ pfexec zlogin zone1
```

Try pinging some hosts on the internet. Remember to update the root password.

Note that the zone's internetworking is possibly subject to external firewalls on your LAN, and/or access to a proxy server, etc.

Snapshot the zone

You might want to clone another zone from these presets along with working networking:

```
$ pfexec zlogin zone1 init 5
$ pfexec zfs snapshot -r rpool/zones/build/zone1@initialNetSetup
$ pfexec zoneadm -z zone1 boot
```

Prepare the compilation environment

You can follow the illumos and OpenIndiana subproject guides on setting up the recommended environments (compilers, source code repositories, etc.):

- Building with oi-build
- How To Build Illumos (and its child pages)
- Working on several bugs at once
- *other links welcome*

You might want to delegate access to common source-code workspaces (i.e. by `lofs`-mounting them from GZ into LZs), to your private package depots, etc., for example:

```
$ pfexec zonecfg -z zone1
add fs
set dir=/export/home
set special=/export/home
set type=lofs
add options nodevices
end
verify
commit
exit

$ pfexec mkdir /zones/build/zone1/root/export/home
$ pfexec ln -s ./export/home/illumos-dev/code /zones/build/zone1/root/code
```

- NOTE: In order to actually mount the `lofs`-mounts you have to use the `mount -F lofs ...` command manually from the global zone, or just reboot the local zone.

Finally, you'd likely want to define the build-user account in the local zone, perhaps using his common home directory from the global zone via `lofs`-mounting or NFS client and `automounter`. For that user you may want to define the `sudo` access rules and/or RBAC to elevate privileges, perhaps to install the built software, etc. (see HOW-TO Setup referential build zone for OpenIndiana Addon Consolidations for more details on that).

Snapshot the zone

You might want to clone another zone from these presets now, so as to instantly start working in the clone:

```
$ pfexec zlogin zone1 init 5
$ pfexec cp -pf /etc/zones/zone1.xml /zones/build/zone1
$ pfexec zfs snapshot -r rpool/zones/build/zone1@initialDevelSetup
$ pfexec zoneadm -z zone1 boot
```

Creating the zone's clones now would involve cloning of the prepared zoneroot's snapshot, copying of the zone configuration (with `zoneadm`, or by copying and modifying zone-description XML files and modifying the `index` file in `/etc/zones` for the time being), and possibly updating the static networking configuration inside the zone root – such as `/etc/hosts`, `/etc/nodename` files at least (also maybe `/etc/hostname.vnicN` for exclusive VNICs and `/etc/motd` to describe this zone's purpose).

Note that there are other ways to clone zones, and unlike the hacks in previous paragraph they are "supported", but those would usually clone a zone's current state instead of using a "golden image" as you can do with the snapshots above. Alternately, you can hold the unused preconfigured "dummy" zone as a golden image, and properly clone it with the supported system tools and methods (making the first clone now for your actual development work).