Productionising ROS when you have no choice (with Bazel)

Ricardo Delfin

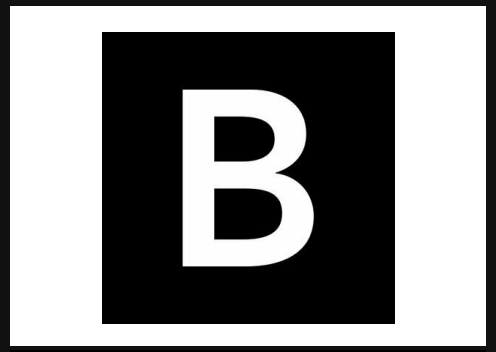
About Me

Facebook - Production Engineer

- Server Provisioning
- Web Monitoring
- Incident Management

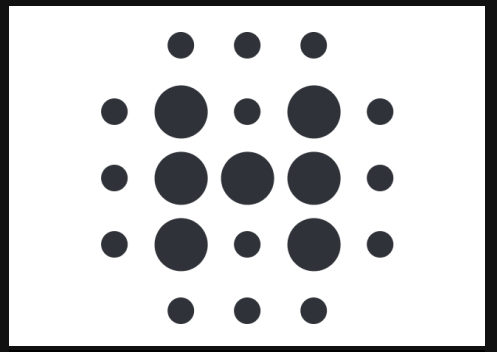
Wayve - Robotics Software Engineer

- Sensor Integration
- Overall Software Integration
- Monitoring



Bloomberg - Software Engineer

- Kernel Deployment
- Observability
- Performance Investigation

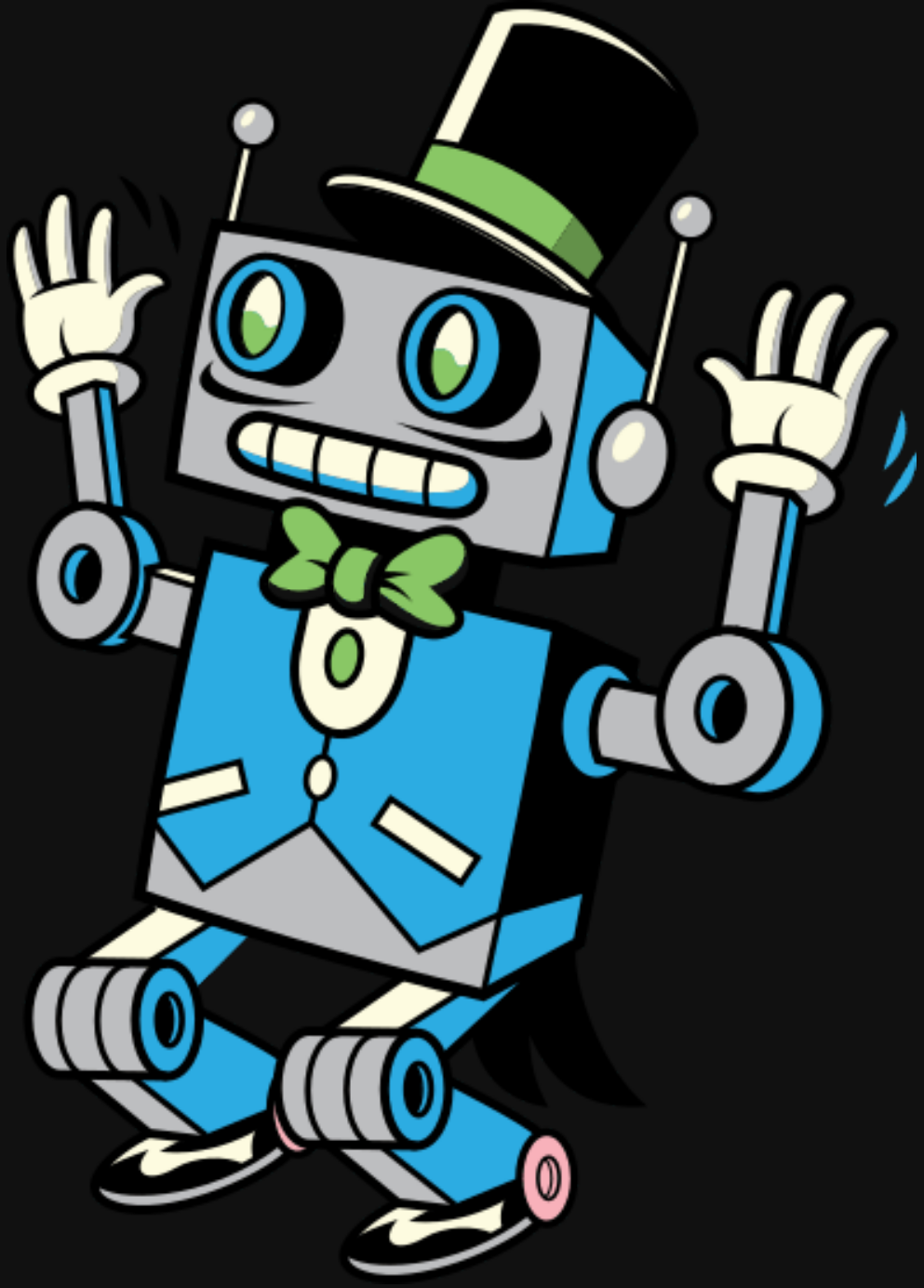


Humanoid - Robotics Software Engineer

- System Integration
- Data Collection Devices
- Deployment

Why ROS?

- Standard for robotics research
- Quick to setup
- Widespread robot support
- Lots of out-of-the-box hardware support
- Easily extensible
- Composable



Source: https://www.ros.org/

Why not ROS?

- Primary deployment by dpkg
- No obvious boot-time launch
- Colcon
- Tied to specific Ubuntu releases
- Often non-deterministic
- Illusion of repeatability
- rosmsg aren't backwards compatible
- Show-stopping bugs in the shared memory topic interface
- Good lord so many timing bugs

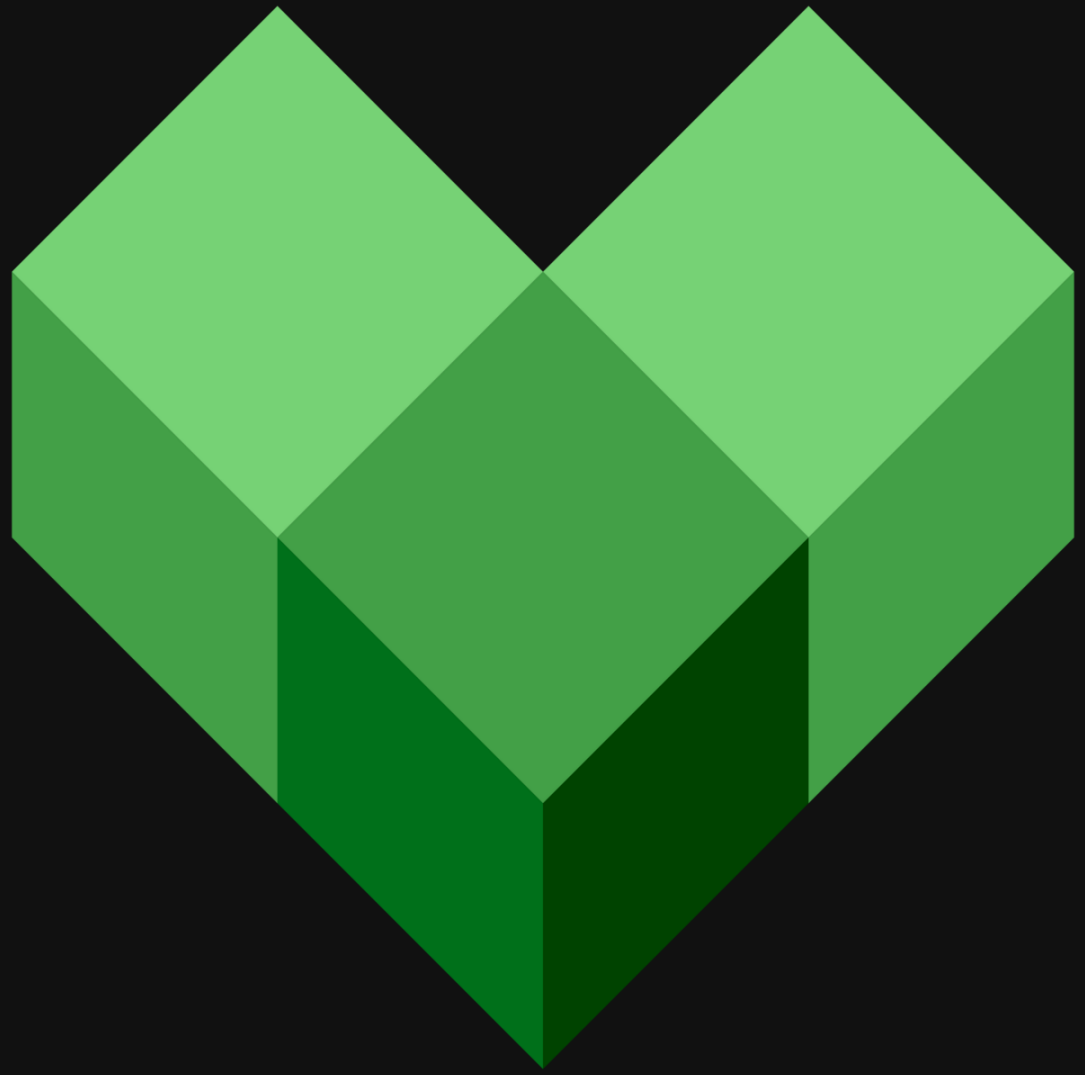
Source: https://jellycat.com/timmy-turtle/

But what if I have to?

- You want to get:
  - Clear dependency definitions
  - Build isolation (hermetic builds)
  - Portability
  - Reproducibility
  - Cheap, fast builds (caching)

But what if I have to?

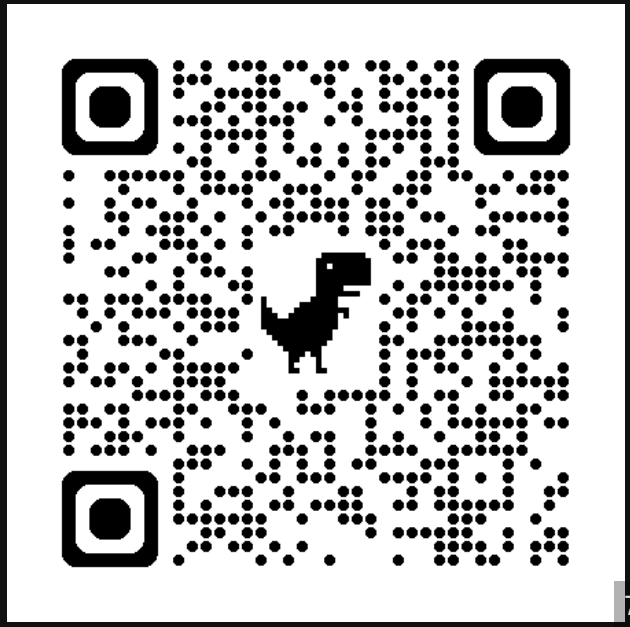
- You want to get:
  - Clear dependency definitions
  - Build isolation (hermetic builds)
  - Portability
  - Reproducibility
  - Cheap, fast builds (caching)
- **Bazel!**



Source: https://bazel.build/

Examples:

https://github.com/rdelfin/ros2_bazel_examples



Building ROS

- https://github.com/mvukov/rules_ros2/
  - Builds ROS from source
  - Fully featured
  - Self-contained runtime (few system libs)
  - Pinned ROS version
  - Independent from Linux distro

Source: https://www.flickr.com/photos/17739115@N00/9900702084

Sample Node Build

```
ros2_cpp_binary(
    name = "talker",
    srcs = ["talker.cpp"],
    deps = [
        "@fmt",
        "@ros2_rclcpp//:rclcpp",
        "//interface:cpp_rdelfin_msgs",
    ],
)

py_binary(
    name = "listener",
    srcs = ["listener.py"],
    deps = [
        "@ros2_rclpy//:rclpy",
        "//interface:py_rdelfin_msgs",
    ],
)
```

ros2_bazel_examples:nodes/BUILD

```
from launch import LaunchDescription
from launch.actions import (
    RegisterEventHandler,
    Shutdown,
)
from launch.event_handlers import OnProcessExit
from launch_ros.actions import Node


def generate_launch_description():
    return LaunchDescription([
        Node(
            name="talker_node",
            executable="nodes/talker",
        ),
        Node(
            name="listener_node",
            executable="nodes/listener",
        ),
    ])
```

ros2_bazel_examples:nodes/nodes.launch.py

Sample Node Build

```
ros2_cpp_binary(
    name = "talker",
    srcs = ["talker.cpp"],
    deps = [
        "@fmt",
        "@ros2_rclcpp//:rclcpp",
        "//interface:cpp_rdelfin_msgs",
    ],
)

py_binary(
    name = "listener",
    srcs = ["listener.py"],
    deps = [
        "@ros2_rclpy//:rclpy",
        "//interface:py_rdelfin_msgs",
    ],
)
```

ros2_bazel_examples:nodes/BUILD

```
from launch import LaunchDescription
from launch.actions import (
    RegisterEventHandler,
    Shutdown,
)
from launch.event_handlers import OnProcessExit
from launch_ros.actions import Node


def generate_launch_description():
    return LaunchDescription([
        Node(
            name="talker_node",
            executable="nodes/talker",
        ),
        Node(
            name="listener_node",
            executable="nodes/listener",
        ),
    ])
```

ros2_bazel_examples:nodes/nodes.launch.py

Sample Node Build

```
ros2_launch(
    name = "launch_nodes",
    launch_file = "nodes.launch.py",
    nodes = [
        ":talker",
        ":listener",
    ],
)
```

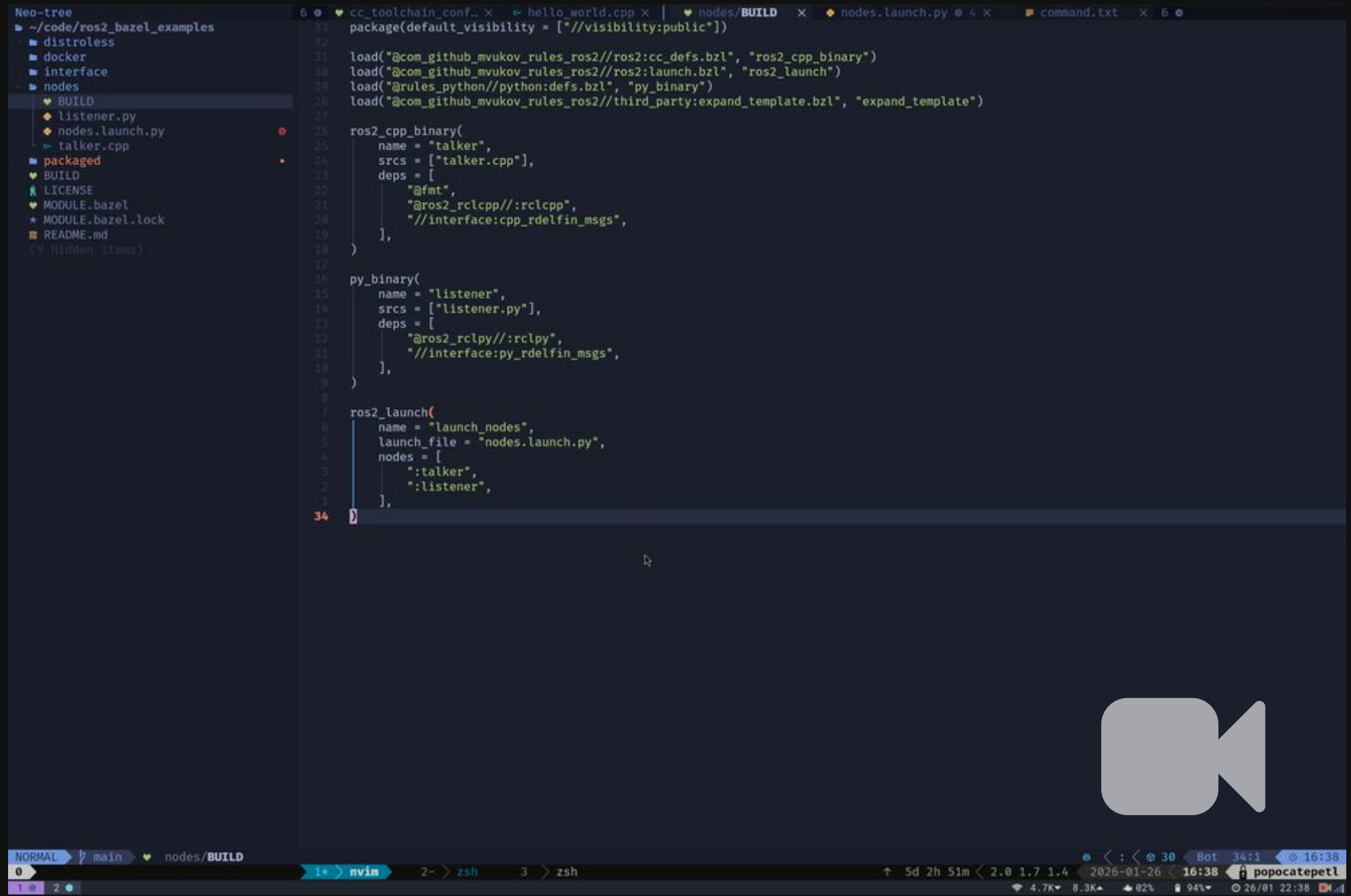
ros2_bazel_examples:nodes/BUILD

```
from launch import LaunchDescription
from launch.actions import (
    RegisterEventHandler,
    Shutdown,
)
from launch.event_handlers import OnProcessExit
from launch_ros.actions import Node


def generate_launch_description():
    return LaunchDescription([
        Node(
            name="talker_node",
            executable="nodes/talker",
        ),
        Node(
            name="listener_node",
            executable="nodes/listener",
        ),
    ])
```

ros2_bazel_examples:nodes/nodes.launch.py

BUILD


Neo-tree
~/code/ros2_bazel_examples
distroless
docker
interface
nodes
BUILD
listener.py
nodes.launch.py
talker.cpp
packaged
BUILD
LICENSE
MODULE.bazel
MODULE.bazel.lock
README.md
cc_toolchain_conf…
hello_world.cpp
nodes/BUILD
nodes.launch.py
command.txt
package(default_visibility = ["//visibility:public"])
load("@com_github_mvukov_rules_ros2//ros2:cc_defs.bzl", "ros2_cpp_binary")
load("@com_github_mvukov_rules_ros2//ros2:launch.bzl", "ros2_launch")
load("@rules_python//python:defs.bzl", "py_binary")
load("@com_github_mvukov_rules_ros2//third_party:expand_template.bzl", "expand_template")
ros2_cpp_binary(
name = "talker",
srcs = ["talker.cpp"],
deps = [
"@fmt",
"@ros2_rclcpp//:rclcpp",
"//interface:cpp_rdelfin_msgs",
],
)
py_binary(
name = "listener",
srcs = ["listener.py"],
deps = [
"@ros2_rclpy//:rclpy",
"//interface:py_rdelfin_msgs",
],
)
ros2_launch(
name = "launch_nodes",
launch_file = "nodes.launch.py",
nodes = [
":talker",
":listener",
],
)
NORMAL
main
nodes/BUILD
Bot 34:1
16:38
nvim
zsh
zsh
5d 2h 51m
2.0 1.7 1.4
2026-01-26
16:38
popocatepetl

Deployment - Container

- Deploy ROS nodes as container
  - Quick to setup
  - Isolated environment
  - Widely supported
  - Can be setup at boot
- https://github.com/bazel-contrib/rules_oci
- https://github.com/bazel-contrib/rules_distroless

Source: https://www.flickr.com/photos/14445655@N04/1494590209

Docker: Code

```
pkg_tar(
    name = "nodes_runfiles_tar",
    srcs = [
        "//nodes:talker",
        "//nodes:listener",
        "//nodes:launch_nodes",
    ],
    package_dir = "/usr/lib/ros_src",
    include_runfiles = True,
)

oci_image(
    name = "app_image",
    base = "@ros_base",
    workdir = "/usr/lib/ros_src/launch_nodes.runfiles/_main",
    cmd = "command.txt",
    tars = [
        ":python_runtime_tar",
        ":nodes_runfiles_tar",
    ],
)

oci_load(
    name = "load_app",
    image = ":app_image",
```

ros2_bazel_examples:docker/BUILD

Docker: Code

```
oci_image(
    name = "app_image",
    base = "@ros_base",
    workdir = "/usr/lib/ros_src/launch_nodes.runfiles/_main",
    cmd = "command.txt",
    tars = [
        ":python_runtime_tar",
        ":nodes_runfiles_tar",
    ],
)

oci_load(
    name = "load_app",
    image = ":app_image",
    repo_tags = ["app:latest"],
)
```

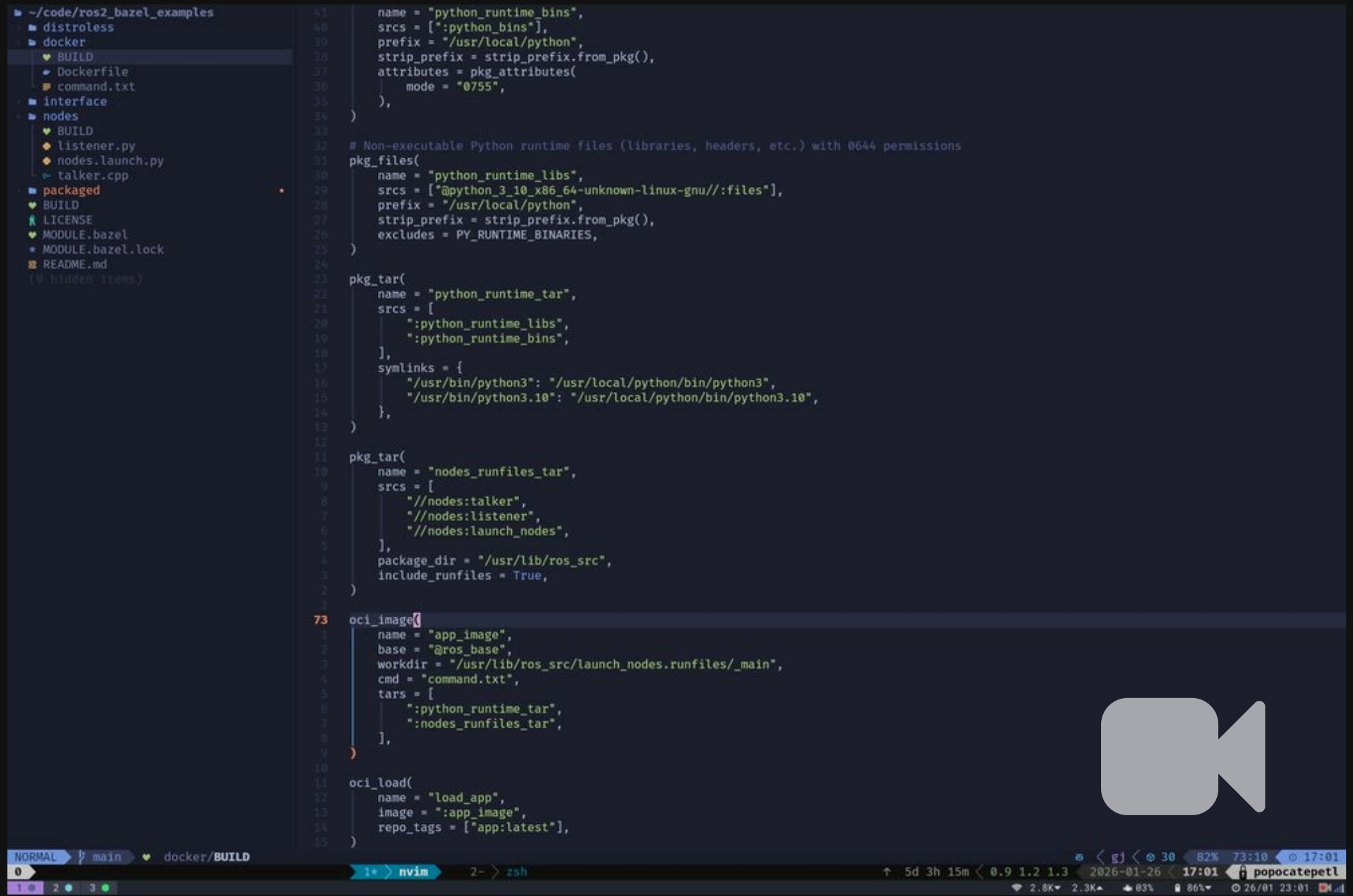
ros2_bazel_examples:docker/BUILD

Docker: Code

```
oci_image(
    name = "app_image",
    base = "@ros_base",
    workdir = "/usr/lib/ros_src/launch_nodes.runfiles/_main",
    cmd = "command.txt",
    tars = [
        ":python_runtime_tar",
        ":nodes_runfiles_tar",
    ],
)

oci_load(
    name = "load_app",
    image = ":app_image",
    repo_tags = ["app:latest"],
)
```

ros2_bazel_examples:docker/BUILD


~/code/ros2_bazel_examples
distroless
docker
BUILD
Dockerfile
command.txt
interface
nodes
BUILD
listener.py
nodes.launch.py
talker.cpp
packaged
BUILD
LICENSE
MODULE.bazel
MODULE.bazel.lock
README.md
(9 hidden items)
name = "python_runtime_bins",
srcs = [":python_bins"],
prefix = "/usr/local/python",
strip_prefix = strip_prefix.from_pkg(),
attributes = pkg_attributes(
mode = "0755",
),
)
Non-executable Python runtime files (libraries, headers, etc.) with 0644 permissions
pkg_files(
name = "python_runtime_libs",
srcs = ["@python_3_10_x86_64-unknown-linux-gnu//:files"],
prefix = "/usr/local/python",
strip_prefix = strip_prefix.from_pkg(),
excludes = PY_RUNTIME_BINARIES,
)
pkg_tar(
name = "python_runtime_tar",
srcs = [
":python_runtime_libs",
":python_runtime_bins",
],
symlinks = {
"/usr/bin/python3": "/usr/local/python/bin/python3",
"/usr/bin/python3.10": "/usr/local/python/bin/python3.10",
},
)
pkg_tar(
name = "nodes_runfiles_tar",
srcs = [
"//nodes:talker",
"//nodes:listener",
"//nodes:launch_nodes",
],
package_dir = "/usr/lib/ros_src",
include_runfiles = True,
)
oci_image(
name = "app_image",
base = "@ros_base",
workdir = "/usr/lib/ros_src/launch_nodes.runfiles/_main",
cmd = "command.txt",
tars = [
":python_runtime_tar",
":nodes_runfiles_tar",
],
)
oci_load(
name = "load_app",
image = ":app_image",
repo_tags = ["app:latest"],
)
NORMAL main docker/BUILD
1* nvim 2- zsh
5d 3h 15m 0.9 1.2 1.3 2026-01-26 17:01 popocatepetl

Deployment - Packages

- DPKG/RPM Solution
  - Installs on the base system
  - Requires a solution for runfiles
  - Requires service deployment (e.g. systemd)
  - Easily deployable!
- https://github.com/bazelbuild/rules_pkg

Source: https://www.flickr.com/photos/51526368@N03/5116363810

Deployment - Packages

```
load("@rules_pkg//pkg:deb.bzl", "pkg_deb")
load("@rules_pkg//pkg:mappings.bzl", "pkg_files")
load("@rules_pkg//pkg:tar.bzl", "pkg_tar")


pkg_files(
    name = "ros_nodes_service",
    srcs = ["ros_nodes.service"],
    prefix = "usr/lib/systemd/system",
)

pkg_tar(
    name = "launch_nodes_tar",
    srcs = [":ros_nodes_service"],
    symlinks = {
        "/usr/bin/launch_nodes": "/usr/lib/ros_src/launch_n
    },
    deps = ["//docker:nodes_runfiles_tar"],
)
```

ros2_bazel_examples:packaged/BUILD

```
[Unit]
Description=ROS Nodes

[Service]
ExecStart=/usr/bin/launch_nodes
WorkingDirectory=/usr/lib/ros_src/launch_nodes.runfiles/_ma
Environment=HOME=/root

[Install]
WantedBy=multi-user.target
```

ros2_bazel_examples:packaged/ros_nodes.service

Deployment - Packages

```
pkg_deb(
    name = "launch-nodes-deb",
    architecture = select({
        "@platforms//cpu:arm64": "arm64",
        "@platforms//cpu:x86_64": "amd64",
    }),
    data = ":launch_nodes_tar",
    description = "Launch Nodes",
    maintainer = "Ricardo Delfin",
    package = "launch-nodes",
    postinst = "postinst.launch_nodes",
    prerm = "prerm.launch_nodes",
    version = "0.1.0",
)
```

ros2_bazel_examples:packaged/BUILD

```
[Unit]
Description=ROS Nodes

[Service]
ExecStart=/usr/bin/launch_nodes
WorkingDirectory=/usr/lib/ros_src/launch_nodes.runfiles/_ma
Environment=HOME=/root

[Install]
WantedBy=multi-user.target
```

ros2_bazel_examples:packaged/ros_nodes.service

Deployment - Images

- Sysroot and image-based deployment
  - Converges system and build environment
  - Avoids dynamic linker issues
  - Avoids differences in different environments
  - Lets you setup remote caching
- Many tools for this:
  - yocto
  - mmdebstrap
  - rules_distroless
  - ...

Source: https://www.flickr.com/photos/75243940@N00/859194667

Deployment - Images

```
load("@rules_pkg//pkg:tar.bzl", "pkg_tar")

pkg_tar(
    name = "flat_rootfs",
    deps = [
        "//distroless:sh",
        "//distroless:passwd",
        "//distroless:group",
        "@noble//:noble",
        "//docker:python_runtime_tar",
        "//docker:nodes_runfiles_tar",
    ],
)
```

ros2_bazel_examples:image/BUILD

OTA Updates

- Make sure you have:
  - Deployment capabilities
  - Monitoring and health-checking
  - Remote access (if security allows)
- https://mender.io/
  - Extensible
  - Wide set of features
  - Solid fleet management

Source: https://github.com/mendersoftware

Thank You!

https://github.com/rdelfin/ros2_bazel_examples

https://github.com/rdelfin

me@rdelfin.com