**Linux**

Micro-Services:

A microservices architecture is an approach to developing a single application as a suite of small services. Each service runs in its own process and communicates with other services through lightweight mechanisms. This approach breaks down a monolithic application into smaller services, where each service serves a single purpose and is deployed as a container. **By porting your** .NET Framework applications to .NET Core or .NET 6, you can run your applications on multiple platforms, reduce your license costs, increase performance, and improve scalability.

* If you want complete control over the configuration of your compute environment, including memory and storage settings, and control over operating system patches: deploy your application as a Linux container on an EC2 instance.

Folder Modular Structure:

**Filesystem Hierarchy Standard** (**FHS**) defines the directory structure and directory contents in Linux distributions. It is maintained by the Linux Foundation

Oracle's BC4J delivers reusable components to the Linux platform

Talk to recycling experts, and they'll speak about the "inherent economy" -- that is, reusing existing materials instead of creating new materials from scratch. The same principle applies to application development. A programming tool that allows you to build reusable modules and business components can dramatically reduce your development times, letting you bring new applications to market faster.

That's the kind of tool Oracle's Business Components for Java (BC4J) is. Recently ported to Linux, BC4J offers a standards-based, server-side Java and XML framework for developers who must build and deploy reusable business components for high-performance Internet applications, such as e-commerce and business-to-business systems. In our tests, BC4J acquitted itself nicely, proving itself a powerful ally to Linux developers who must create n-tier Internet applications and connect them to Oracle databases.

Using BC4J to create reusable business components can reduce development time, allowing you to bring applications to market faster. You can also introduce efficiencies to your programming team by creating libraries of objects once other developers can repurpose those objects in similar applications.

A well-integrated part of Oracle's JDeveloper IDE, BC4J brings Linux developers in line with their Solaris and Windows counterparts. Applications can be executed in either the Oracle Application Server or Oracle8i as EJB, CORBA, or JSP components.

Applications created using BC4J are comprised of five basic framework components: entity objects, associations, view objects, view links, and application modules. Each component is interrelated, meaning you can establish views into database tables and join, filter, and sort data as needed.

Using the Business Components wizard in JDeveloper, we tested BC4J by automatically generating all the components needed to create a simple order entry system. We could then quickly repurpose several of the entity objects to create a simple RMA (return to manufacturer authorization) system supporting order returns. We were impressed by how quickly we could access backend data stores, as well as BC4J's built-in validation capabilities (which let us establish rules to prevent users from entering invalid values into the database).

Each component we created was represented by an XML file and one or more Java files. As you might expect, the XML files store metadata, while the Java files store the object code that facilitates an application's actions. Java developers will quickly recognize that each BC4J component is organized using familiar directory-based semantics. Ultimately, this type of separation greatly enhances your ability to customize and extend business logic, without forcing you to focus on code changes.

Once created in the BC4J editor, applications can be deployed as either CORBA server objects or Enterprise JavaBeans (EJB) session beans on any server platforms that support Java. Using the included testing tool, we could quickly put our business logic through its paces within the development environment. You get several testing options; for instance, you can deploy the component as a CORBA or EJB object inside Oracle8i, or you can simply connect to the Application module locally.

Based on our tests and the tight integration between BC4J, JDeveloper, and Oracle's flagship database products (BC4J is an integral part of Oracle's JDeveloper IDE, which, in turn, is part of the overarching Internet Development Suite), it's clear that BC4J would be most effective in an all-Oracle environment. But even if that's not the case, Linux developers looking to level the playing field with their Windows and Solaris counterparts would do well to take BC4J out for a test run.

Security:

Vault is an open-source tool that provides a secure, reliable way to store and distribute secrets like API keys, access tokens, and passwords. Software like Vault can be critically important when deploying applications that require the use of secrets or sensitive data (https://www.digitalocean.com/community/tutorials/how-to-securely-manage-secrets-with-hashicorp-vault-on-ubuntu-16-04)Dm-crypt is **a Linux kernel-level encryption mechanism that allows users to mount an encrypted file system**. Mounting a file system is the process in which a file system is attached to a directory (mount point), which makes it available to the operating system.

A daemon is **a service process that runs in the background and supervises the system or provides functionality to other processes**. Traditionally, daemons are implemented following a scheme originating in SysV Unix

Linux FireWalls:

A Linux firewall is defined as a solution or service that regulates, protects, and blocks network traffic as it passes to and from a Linux-based environment. Given that nearly 75% of the world’s servers run on Linux, these solutions are essential to provide secure access to users and end customers

Endian Firewall Community (EFW) is a turnkey or ready-to-use [security solution](https://www.spiceworks.com/tech/cloud/articles/top-10-hybrid-cloud-security-solution-companies/)built on Linux. It requires a hardware shell or virtualized environment to reside and offers protection for Linux-based environments of various sizes. You can also download a free, limited version of EFW as software installed on your existing Linux PC. (<https://www.spiceworks.com/it-security/network-security/articles/top-10-linux-firewall-solutions/>)

(https://linux-audit.com/gdpr™-compliance-technical-requirements-for-linux-systems/)

# **Checking the Linux Kernel with Static AnalySparse, Smatch, and Coccinelle**

# **Two other must-use Linux kernel static analysis tools are Sparse and Smatch. Sparse was written by Linus Torvalds. It’s a simple C parser that can view a program’s structure and create a symbol table showing exactly where every global symbol is defined.**

# **Sparse, however, only looks at a program’s local code. Smatch lets you see how values change across a sequence of code. It also enables you to detect conditions that will always, or never, be true; null pointers; and locks that end up in different states depending on the code path. Needless to say, this can be very helpful for validating error paths and rarely tested code.**

# **“Sparse and Smatch are designed with kernel in mind and have hooks to run from the kernel Makefile,” Khan said. “They are easy to incorporate into the development workflow and patch-acceptance workflow. Sparse and Smatch have external dependencies but they are easier to manage and maintain these tools on development and test systems.”**

# **Another important tool, which is no longer Linux kernel-specific, is Coccinelle. This is a pattern-matching and text-transformation tool that can analyze complex, tree-wide patches and detect problematic programming patterns. It works by applying semantic patches via the top-level Makefile. Typically it delivers a report to the developer, but you can run it to produce proposed patches for the problem it encounters.**

# **Khan thinks it’s “a powerful and complex tool. However, it suffers from a steep learning curve to use effectively and [it’s] a bit hard to incorporate into the development and patch acceptance workflow.” Therefore, she favors Sparse and Smatch.**

# **Still, developers agree that Cocinelle is worth the time and trouble to master. For examples on how to use it, check out Julia Lawall’s “Coccinelle: 10 Years of Automated Evolution in the Linux Kernel” presentation.**

# **Khan pointed to gcc and clang as tools outside the kernel that are easier to use as part of the kernel-development workflow. Specifically, Khan recommends gcc 10 because of the new -fanalyzer option. This is a built-in C static analysis tool. It’s not yet available for C++.**

# **She likes it because it’s “helping find problems in the kernel code. This feature is useful because it added support for detecting [the] Common Weakness Enumeration Software Development category.” This category includes such mistakes as null-dereference and use-after-free, among others.**

# **Khan continued, “I am excited to see gcc adding -fanalyzer support and would like to see increased static analysis coverage to detect Common Weakness Enumeration (CWEs) that pertain to software development.” This, in turn, “will make it easier for Linux and its ecosystem to support the needs of safety-critical domains that require/enforce static analysis.”**

# **sis Tools**